TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

TUUKKA VAINIO

INERTIAMITTAUSJÄRJESTELMÄ

Diplomityö

Tarkastajat: TkT Jussi Collin, prof. Jarmo Takala
Tarkastajat ja aihe hyväksytty
Tieto- ja sähkötekniikan tiedekuntaneuvoston kokouksessa 17.08.2016

TIIVISTELMÄ

TUUKKA VAINIO: Inertiamittausjärjestelmä
Tampereen teknillinen yliopisto
Diplomityö, 53 sivua, 9 liitesivua
Joulukuu 2016
Tietotekniikan koulutusohjelma
Pääaine: Pervasive Systems
Tarkastajat: TkT Jussi Collin, prof. Jarmo Takala
Avainsanat: inertiamittausyksikkö, satelliittipaikannus, Raspberry Pi, Bluetooth

Mikroelektromekaanisten (engl. Micro Electrical Mechanical System, MEMS) inertia-antureiden käyttöä usein rajoittaa niiden suuri mittausvirhe. Mittausvirheiden analysointia varten inertia-antureilla tehtävissä mittauksissa olisi hyvä olla mukana tarkka referenssipiste, jonka avulla heikompien antureiden mittausdataa voidaan korjata jälkiprosessoinnin yhteydessä. Referenssipisteen täytyy olla merkittävästi mittauksissa käytettyjä inertia-antureita tarkempi, jotta niiden kalibrointi olisi mahdollista.

Tässä työssä suunnitellaan ja toteutetaan referenssimittalaitteisto, jota voidaan käyttää halvempien MEMS-antureiden virheiden analysointiin ja kalibrointiin. Laitteistoon kuuluu inertiamittausyksikkö, joka on riittävän tarkka havaitsemaan maapallon pyörimisliikkeen sekä satelliitti-vastaanotin, jonka avulla järjestelmän sijainti voidaan määrittää alle metrin tarkkuudella. Laitteistoon kuuluu myös ohjausyksikkö, jolle toteutetun ohjelmiston avulla järjestelmää voidaan etäohjata.

Suunniteltua mittauslaitteistoa testattiin erilaisissa ajoneuvoissa, kuten esimerkiksi pienmetsäkoneessa. Tämän lisäksi järjestelmä inertiamittausyksikön tarkkuus osoitettiin toteuttamalla pohjoisenhaku-sovellus, joka havaitsee napapohjoisen suunnan maapallon pyörimisliikkeen perusteella.

ABSTRACT

TUUKKA VAINIO: Inertial Measurement System
Tampere University of Technology
Diplomityö, 53 pages, 9 Appendix pages
December 2016
Master's Degree Programme in Information Technology
Major: Pervasive Systems
Examiner: Dr.Sc. Jussi Collin and Prof. Jarmo Takala
Keywords: inertial measurement unit, satellite positioning, Raspberry Pi, Bluetooth

The most common problem of the microelectromechanical (MEMS) inertial sensors is their relatively large measurement error. The error is the reason why there should be a high precision reference device which can be used to identify the error components. The reference can be used to correct the errors of lower grade sensors when post-processing the collected data. In order to measure errors in field tests, the reference point should be significantly more accurate than sensors used in the measurement.

In this thesis a reference measurement system is designed and implemented. The reference system has an inertial measurement unit (IMU) accurate enough to detect the earth's rotation rate and a satellite receiver which can locate the position of the system with precision of less than one meter. The system also has a control unit which controls the system.

The measurement system was tested with a small forestry machine. The precision of the IMU was demonstrated by implementing a graphical north finding application which can locate the true north by detecting earth's rotation rate.

ALKUSANAT

Tämä diplomityö on tehty Tampereen teknillisen yliopiston tietotekniikan laitokselle osana TEKES-rahoitettua Living Lab Bus -projektia. Haluan kiittää työn tarkastajia Jussi Collinia ja Jarmo Takalaa mahdollisuudesta työn tekemiseen.

Kiitokset myös Olli Useniukselle ja Usewood Forest Tec Oy:lle mahdollisuudesta käydä Muuramessa katsomassa ja mittaamassa kuuluisaa metsäkonettanne. Toivottavasti metsäkone ei jatkossakaan juutu ojaan ja maailmanvalloitus etenee suunnitellusti!

Lisäksi haluan kiittää perhettä ja ystäviä tuesta ja kannustuksesta. Erityiset kiitokset Tuomakselle, Suville ja Päiville työn kommentoinnista.

Tampereella, 22.11.2016

Tuukka Vainio

SISÄLLYS

KUVALUETTELO

TAULUKKOLUETTELO

LISTAUSLUETTELO

LYHENTEET JA MERKINNÄT

ARM	Edistynyt RISC prosessoriarkkitehtuuri (Advanced RISC Machines)
DFT	Diskreetti Fourier'n muunnos (Discrete Fourier Transform)
ECEF	Maapalloon sidottu koordinaatisto (Earth-Centered Earth-Fixed)
FOG	Kuituoptinen gyroskooppi (Fiberoptic Gyroscope)
FTP	Tiedostonsiirtoprotokolla (File Transfer Protocol)
GLONASS	Venäläinen maailmanlaajuinen navigointisatelliittijärjestelmä (Globalnaya navigatsionnaya sputnikovaya sistema)
GNSS	Maailmanlaajuinen paikannusjärjestelmä (Global Navigation Satellite System). Yleisnimitys kaikille satelliittipaikannusjärjestelmille
GPS	Yhdysvaltalainen maailmanlaajuinen paikannusjärjestelmä (Global Positioning System)
I^2C	Inter-Integrated Circuit
IMU	Inertiamittausyksikkö (Inertial Measurement Unit)
INS	Inertianavigointi järjestelmä (Inertial Navigation System)
IoT	Esineiden ja asioiden internet (Internet of things)
JSON	Javascriptiin perustuva kevyt ja alustariippumaton tiedon esitysmuoto (JavaScript Object Notation)
MEMS	Mikrosysteemi tai mikromekaaninen systeemi (Micro Electrical Mechanical System)
RFCOMM	Bluetooth protokollapinoon kuuluva tiedonsiirtoprotokolla.
RISC	Suoritinarkkitehtuurin suunnitteluperiaate, jossa konekieliset käskyt pyritään pitämään yksinkertaisina (Reduced Instruction Set Computer)
RS232	Sarjaliikennestandardi, joka määrittelee päätelaitteen (DTE) ja verkkopäätteen (DCE) välisen kommunikaation.
RS422	Sarjaliikennestandardi, joka RS232 poiketen käyttää differentiaalista signalointia.
SBAS	Satelliittipohjainen parannusjärjestelmä (Satellite Based Augmentation System)
SDP	Bluetooth-palvelun löytämisprotokolla (Service Discovery Protocol)
SPI	Serial Peripheral Interface
SSH	Tiedonsiirtoprotokolla, jolla taataan turvallinen yhteys turvattoman verkon ylitse (Secure Shell)

TTFF	Ensimmäiseen paikkaratkaisuun kuluva aika (Time To First Fix)
TTY	Unixia ohjaava terminaali tai prosessi. Lyhenne tulee englannin kielen sanasta *teletypewriter*
UART	Universaali asynkroninen vastaanotin/lähetin (Universal asynchronous receiver/transmitter)
UML	Object Management Groupin standardoima graafinen mallinnuskieli (Unified Modelling Language)
USB	Universaali sarjaväylä (Universal Serial Bus)
WGS 84	Maailmanlaajuinen koordinaattijärjestelmä (World Geodetic System)

δt	Kellopoikkeama
λ	Leveysaste WGS 84 -koordinaatistossa (Hervanta $61,450°\ N$)
ρ	Pseudoetäisyys
c	Valonnopeus tyhjiössä (299 792 458 m/s)
g_n	Normaaliputoamiskiihtyvyys ($9,80665\ m/s^2$)
$\bar{m}_w$	Gyroskooppi-mittauksen normaalivektori
$\bar{m}_a$	Kiihtyvyysanturi-mittauksen normaalivektori
r	Etäisyys

1. JOHDANTO

Inertiamittausyksikkö (engl. Inertial Measurement Unit, IMU) on laite, joka mittaa kappaleen lineaarista itseiskiihtyvyyttä (engl. specific-force) ja kulmanopeutta kiihtyvyysantureiden ja gyroskooppien avulla. Varsinkin MEMS-teknologian (engl. Micro Electrical Mechanical System) kehittymisen myötä inertia-anturit ovat yleistyneet voimakkaasti. Esimerkiksi mobiililaitteet käyttävät inertia-antureita näytön oikean asennon määrittämiseen [5]. MEMS-antureiden isoin ongelma on niiden epätarkkuus verrattuna esimerkiksi ilmailussa käytettyihin inertia-antureihin [5].

MEMS-teknologian lisäksi myös tarkemmat optiset teknologiat ovat tulleet entistä pienemmiksi ja halvemmiksi. Etenkin optisten gyroskooppien hinnan alentuminen mahdollistaa näiden entistä laajemman käytön tieteellisessä tutkimuksessa, sekä kaupallisissa sovelluksissa [5].

Tässä diplomityössä suunniteltiin ja toteutettiin inertiamittausjärjestelmä, jonka inertiamittausyksikön avulla voidaan kalibroida halvempia ja epätarkempia inertiamittausyksiköitä. Suunniteltua järjestelmää voidaan käyttää inertiamittauksissa tarkkana referenssipisteenä, johon muiden mittauksissa käytettyjen antureiden mittausdataa voidaan verrata. Ajoneuvoissa tehtäviä mittauksia varten järjestelmään liitettiin myös satelliittivastaanotin, jonka avulla järjestelmän sijainti voidaan määrittää alle metrin tarkkuudella.

Diplomityön rakenne on seuraava: luvussa 2 esitellään tarkemmin inertia-antureiden ja satelliittipaikannuksen periaatteita ja teoriaa. Luvun tarkoitus on antaa lukijalle peruskäsitys siitä, mitä inertia-anturit ovat ja miten satelliittipaikannusjärjestelmät (engl. Global Navigation Satellite System, GNSS) toimivat. Tämän lisäksi luvuissa käsitellään myös inertia-antureiden ja satelliittipaikannusjärjestelmien mittaustarkkuutta sekä näiden tyypillisiä virhelähteitä.

Seuraavaksi työssä suunnitellaan inertiamittauslaitteisto ja esitellään sitä ohjaava ohjelmisto. Luvussa 3 vertaillaan järjestelmän vaihtoehtoisia komponentteja ja pe-

rustellaan tehdyt valinnat. Luvussa 4 esitellään tarkemmin edellisessä luvussa valitut komponentit sekä kuvataan näiden yhdistäminen yhdeksi mittauslaitteistoksi. Luvussa 5 esitellään mittauslaitteistoa ohjaavan ohjelmisto.

Työn lopussa esitellään laitteiston erilaisia käyttökohteita. Laitteistoon valitun inertiamittausyksikön tarkkuutta testataan pohjoisenhaku-sovelluksella, jonka avulla pyritään löytämään napapohjoisen suunta maan pyörimisliikkeen perusteella. Lisäksi laitteistoa käytettiin halvempien MEMS-antureiden tukena pienmetsäkoneen tärinää tutkivissa mittauksissa.

2. INERTIAMITTAUSYKSIKÖT JA SATELLIITTIPAIKANNUS

Tässä luvussa esitellään inertiamittausyksiköiden ja satelliittipaikannuksen teoriaa, sekä näiden virhelähteitä.

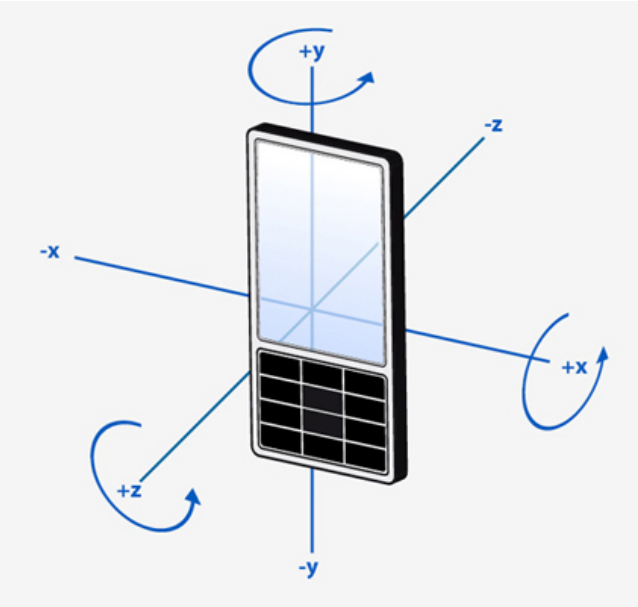
2.1 Inertiamittausyksiköt

Inertiamittausyksikkö on laite, joka mittaa lineaarista itseiskiihtyvyyttä ja kulmaliikettä kolmessa ulottuvuudessa. Mittausyksiköt koostuvat kolmen gyroskoopin- ja kiihtyvyysanturin ryhmistä, joissa kukin anturi mittaa yhtä kolmiulotteisen koordinaatiston akselia (kuva 2.1). Kiihtyvyysantureita ja gyroskooppeja kutsutaan *inertia-antureiksi*, sillä ne mittaavat oman inertiaalikoordinaatistonsa muutoksia, eivätkä ne tarvitse mittauksiinsa ulkoista kiintopistettä. [1]

2.1.1 Gyroskoopit

Gyroskooppi on anturi, joka mittaa kulmanopeutta tietyn akselin suhteen [1]. Kulmanopeuden perusteella gyroskoopilla voidaan määrittää kappaleen asento ja sen muutokset. Koska yksi gyroskooppi mittaa kulmanopeutta vain yhden akselin suhteen, mitattavaan kappaleeseen täytyy kiinnittää kolme gyroskooppia, jotta sen asennon muutoksia voidaan seurata kolmessa ulottuvuudessa. Kuvassa 2.1 akseleiden ympäri kiertyvät nuolet ilmaisevat gyroskooppien mittaamaa kulmanopeutta.

Gyroskooppi on melko vanha keksintö: esimerkiksi 1800-luvun Euroopassa maapallon pyörimisliikettä tutkittiin mekaanisten gyroskooppien avulla [44]. Ensimmäiset gyrokompassit vuorostaan kehitettiin 1900-luvun alussa ja toisen maailmansodan yhteydessä gyroskooppeja käytettiin ilmatorjuntatykkien, lentokoneiden ja V2-ohjusten vakauttamiseen. Maailmansodan jälkeen gyroskooppien tärkeimpiä sovelluksia ovat olleet inertianavigointijärjestelmät (engl. Inertial Navigation System,



Kuva 2.1 *Kolmiulotteisen koordinaatiston akselit. Kiihtyvyysanturit mittaavat kiihtyvyyttä suorien akselin suuntaisesti ja gyroskoopit mittaavat akseleiden ympäri tapahtuvaa kulmanopeutta. Kuva Qt:n anturirajapinnan dokumentaatiosta [37, Qt Sensors].*

INS) [5], joiden avulla navigoitsijan sijainnin muutos pystytään määrittämään pelkästään antureiden mittaaman liikkeen perusteella. Tällaisia järjestelmiä käytetään muun muassa lentokoneissa, laivoissa ja sukellusveneissä.

Nykyään gyroskooppeja on erilaisia ja niiden käyttökohteet vaihtelevat tekniikan antaman tarkkuuden ja anturin hinnan mukaan: tarkimpia ja samalla kalleimpia gyroskooppeja ovat lasergyroskoopit, joita käytetään laivojen, lentokoneiden ja sukellusveneiden paikannusjärjestelmissä [23]. Pieniä ja halpoja MEMS-gyroskooppeja käytetään vuorostaan älypuhelimissa ja tableteissa mm. laitteen asennon ja eri eleiden tunnistamiseen [5]. Pienen kokonsa vuoksi tarkimpia MEMS-gyroskooppeja käytetään lääketieteessä esimerkiksi askeltutkimuksessa, kaatumisen havaitsemisessa ja unitutkimuksessa [9].

2.1.2 Kiihtyvyysanturit

Kiihtyvyysanturi on anturi, joka mittaa kappaleen *itseiskiihtyvyyttä* (engl. specific-force). Itseiskiihtyydellä tarkoitetaan kappaleen kiihtyvyyttä, joka johtuu kappaleen ulkoisista voimista pois lukien painovoima. Toisin sanoen, jos kappale on vapaassa

pudotuksessa, sen itseiskiihtyvyys on nolla. Vastaavasti, jos kiihtyvyysanturi on pöydällä levossa, näkyy pöydästä johtuva tukivoima anturissa painovoiman vastaisena komponenttina. [1] [23]

Kiihtyvyysanturi voi mitata joko lineaaristakiihtyvyyttä tai kulmakiihtyvyyttä, mutta yleensä kiihtyvyysantureilla tarkoitetaan lineaaristakiihtyvyyttä mittaavia antureita [1]. Useimmiten kiihtyvyysanturit eivät käytä yksikkönään m/s^2, vaan niiden perusyksikkönä käytetään normaaliputoamiskiihtyyttä g_n [1][23]. Putoamiskiihtyvyys vaihtelee eri pituuspiireillä, mutta g_n viralliseksi suuruudeksi on määritelty sen arvo 45° pituuspiirillä, eli $9,80665\ m/s^2$ [42]. Samalla tavalla kuin gyroskooppeja, myös kiihtyvyysantureita pitää olla yksi jokaista mitattavaa akselia kohden.

Ensimmäiset kiihtyvyysanturit kehitettiin 1920-luvulla, joten se on hieman gyroskooppia nuorempi keksintö [49]. Ensimmäisiä kiihtyvyysantureita käytettiin erilaisten koneiden tärinän mittaamiseen, mutta toisen maailmansodan jälkeen, myös kiihtyvyysantureita alettiin käyttämään gyroskooppien tavoin inertianavigointijärjestelmissä [5]. Gyroskooppien ja kiihtyvyysantureiden ominaisuudet täydentävät toisiaan, jonka vuoksi ne usein paketoidaankin kolme kiihtyvyysanturia ja gyroskooppia sisältäväksi inertiamittausyksiköksi [23].

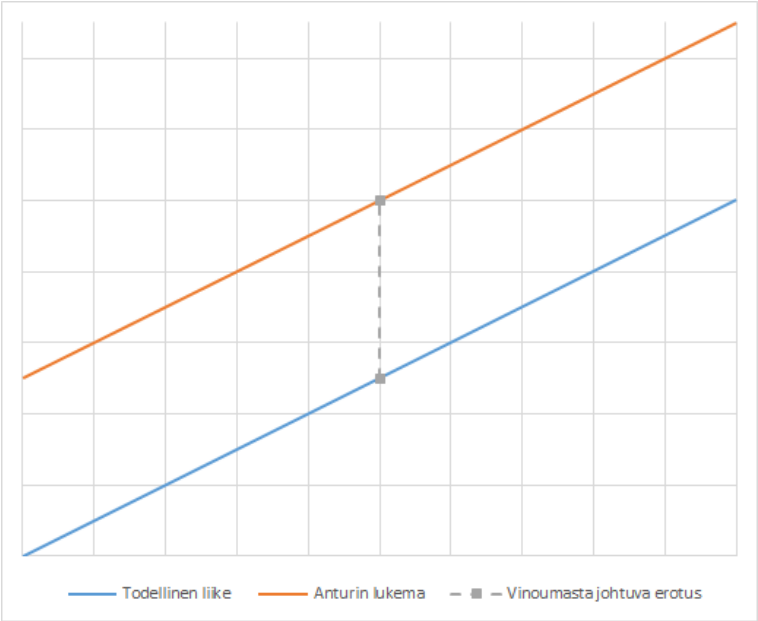
2.1.3 Inertia-antureiden virhelähteet

Tässä luvussa esitellään inertia-antureille yhteiset virhetyypit: vinouma, skaalausvirhe, ristikytkentävirhe ja kohina. Kustakin virhelähteestä johtuva systemaattinen virhe voi vaihdella anturilla suoritettavien ajojen välillä sekä yksittäisen ajon aikana. Lisäksi virheen suuruus vaihtelee anturin lämpötilan mukaan. [23]

Vinouma

Vinouma (engl. bias) on sekä gyroskoopeissa, että kiihtyvyysantureissa esiintyvä vakiovirhe [23] [1]. Vinouman vuoksi anturi antaa nollatasosta poikkeavan lukeman, vaikka se olisikin täysin paikoillaan. Vinouma ei ole riippuvainen anturiin vaikuttavista voimista, eikä anturin liike vaikuta vinouman määrään [23]. Kuvassa 2.2 on esitetty, kuinka vinouma vaikuttaa anturin antamaan lukemaan.

Vinouma on usein suurin yksittäinen inertiamittausyksikön virhelähde ja se jaetaan kahteen osaan: staattiseen ja dynaamiseen komponenttiin. Staattinen komponentti



Kuva 2.2 *Anturin vinouma. Alempi viiva on anturin todellinen liike, mutta vinoumasta johtuvan virheen vuoksi (harmaa) anturi näyttääkin oikeasta liikkeestä poikkeavaa lukemaa (punainen).*

kuvastaa inertiamittausyksikön kaikille antureille yhteistä vinoumaa, joka voi kuitenkin vaihdella eri käyttökertojen välillä. Dynaaminen komponentti vuorostaan kuvastaa yhden käyttökerran aikana tapahtuvaa vinouman vaihtelua ja se sisältää mm. lämpötilasta johtuvan vaihtelun. [23]

Koska vinouma on merkittävin inertiamittausyksikön virhelähde, sen suuruus kertoo paljon mittausyksikön tarkkuudesta. Taulukossa 2.1 on luetteloitu kiihtyvyysantureiden ja gyroskooppien laatuluokituksia ja näiden vinoumia. Taulukossa mainitut laatuluokitukset eivät ole standardoituja ja luokitusten määritelmät vaihtelevatkin todella paljon eri kirjallisuuslähteissä [22][23][44]. Laatuluokitukset antavat kuvan siitä, kuinka tarkkoja antureita eri käyttökohteissa käytetään.

Skaalauskerroinvirhe

Skaalauskerroinvirheellä tarkoitetaan inertiamittausyksikön antaman lukeman ja todellisen liikkeen välistä eroa [1]. Gyroskoopin tapauksessa anturia voidaan kääntää tietyllä nopeudella, mutta skaalauskerroinvirheen vuoksi anturi ei pysty tulkitsemaan liikkeen aiheuttamaa signaalia riittävän tarkasti. Teollisten MEMS-antureiden

Taulukko 2.1 *Inertiamittausyksiköiden laatuluokat sekä näiden tyypilliset vinoumat ja hinnat. [23, p. 138]*

Luokitus	Kiihtyvyysanturi	Gyroskooppi	Hinta
Consumer Grade	$> 50\ mg_n$	$> 100\ ^\circ$/h	< 5 €
Industrial Grade	$10\ mg_n$	10-100 $^\circ$/h	< 10 €
Tactical Grade	$1\ mg_n$	1-10 $^\circ$/h	1600 - 25 000 €
Aviation Grade	$0.03\text{-}0.1\ mg_n$	$< 0.01\ ^\circ$/h	16 000 - 40 000 €
Marine Grade	$0.05\ mg_n$	$< 0.0001\ ^\circ$/h	$\approx$ 800 000 €

skaalauskerroinvirhe on tyypillisesti noin 1-3% luokkaa [28], joten jos tällaista MEMS-gyroskooppia käännetään 90°/s nopeudella, sen lukemassa voi olla noin $\pm\ 2,7^\circ$ verran skaalauskerroinvirhettä.

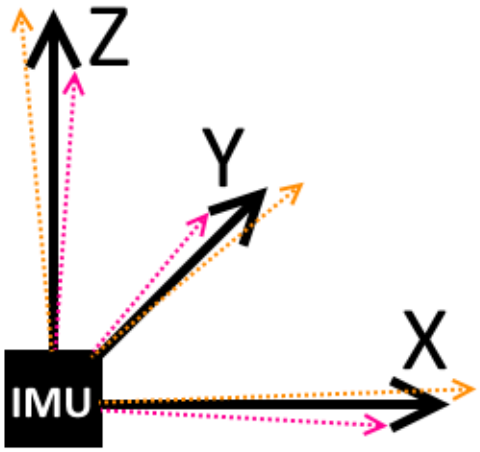
Skaalauskerroinvirheen suuruus on suoraan verrannollinen anturin kokeman liikkeen suuruuteen: gyroskoopin skaalauskerroinvirhe on suoraan verrannollinen anturin pyörimisliikkeen määrään ja kiihtyvyysanturinvirhe on vastaavasti suoraan verrannollinen anturin kokeman ominaiskiihtyvyyden määrään. Mitä nopeammassa kulmaliikkeessä tai mitä suuremmassa kiihtyvyydessä anturit ovat, sen suurempi on skaalauskerroinvirheen osuus antureiden antamasta lukemassa. [23]

Ristikytkentävirhe

Ristikytkentävirhe (engl. cross-coupling error) johtuu antureiden virheellisestä asennuksesta: anturit eivät yksinkertaisesti ole täysin kohtisuorassa toisiaan vasten [1]. Esimerkiksi kuvan 2.3 tapauksessa inertiamittausyksikön anturit eivät ole täysin koordinaattiakseleiden suuntaisia. Ristikytkentävirhe voi johtua esimerkiksi valmistukseen liittyvistä rajoitteista, joiden vuoksi antureiden akseleita ei saada kohdistettua oikein. Toisaalta joskus ristikytkentävirhe on suoraviivaista kompensoida sopivalla korjausmatriisilla. [23]

Kohina

Kohinalla tarkoitetaan kaikenlaisia häiriölähteitä, jotka voivat haitata antureiden mittaustarkkuutta. Kohinan syy vaihtelee erilaisten antureiden välillä. Sähköinen



Kuva 2.3 *Ristikytkentävirhe. Inertiamittausyksikön antureiden pitäisi olla sen koordinaatiston akseleiden suuntaisia (mustat nuolet), mutta ristikytkentävirheen vuoksi näin ei ole. Kuvan tapauksessa anturit (oranssit ja punaiset nuolet) eivät ole inertiamittausyksikön koordinaatiston suuntaisia. Anturit eivät myöskään ole kohtisuorassa toisiinsa verrattuna.*

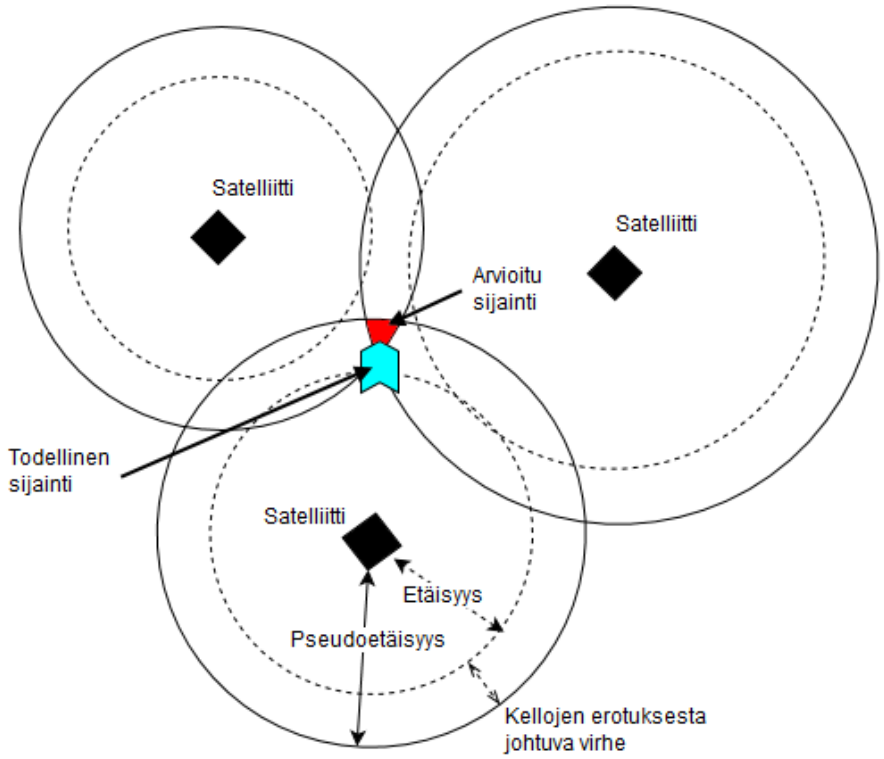
kohina on erittäin suuri häiriölähde MEMS-antureissa, joissa mitattava elektroninen signaali on jo valmiiksi hyvin heikko. Heilureihin perustuvissa kiihtyvyysantureissa kohinaa voi syntyä heilurin mekaanisesta epävakaudesta. Lisäksi joidenkin gyroskooppien normaalista toiminnasta johtuva värinä voi aiheuttaa kohinaa samassa yksikössä oleviin kiihtyvyysantureihin. [23]

2.2 Satelliittipaikannus

Satelliittipaikannuksessa käyttäjän sijainti määritetään maapalloa kiertävien paikannussatelliittien avulla. Navigointisatelliitit ovat usein osa suurempaa globaalia paikannussatelliittijärjestelmää. Järjestelmä voidaan jakaa satelliitteja ohjaavaan *kontrolliosaan*, satelliiteista koostuvaan *avaruusosaan* ja käyttäjän laitteistosta koostuvaan *käyttäjäosaan* (engl. Control Segment, Space Segment ja User Segment). Tällaisia järjestelmiä ovat muun muassa yhdysvaltalainen GPS, venäläinen GLONASS, eurooppalainen Galileo, sekä kiinalainen Beidou. [1] [27]

2.2.1 Paikannus signaalin saapumisajan perusteella

Kaikkien edellä mainittujen satelliittijärjestelmien toiminta perustuu satelliitin lähettämän viestin saapumisajan määrittämiseen (engl. Time of Arrival, TOA). Me-



Kuva 2.4 *Satelliittipaikannuksen periaate: paikantaja (sininen) pystyy määrittämään etäisyytensä kustakin satelliitista viestin lähetysajan ja saapumisajan perusteella (sisemmät ympyrät). Aluksen ja satelliittien kellot eivät kuitenkaan ole samassa ajassa, joten laskettuun etäisyyteen on lisättävä tästä erotuksesta johtuva kellovirhe (uloimmat ympyrät). Aluksen arvioitu sijainti on siis pseudoetäisyyksien muodostaman poikkileikkauksen sisällä (punainen alue). Kuva mukailtu [27, Fig. 2.5] ja [23, Fig. 8.11] pohjalta.*

netelmässä vastaanottaja saa satelliitilta viestin, josta käy ilmi satelliitin sijainti ja satelliitin atomikellon näyttämä aika. Menetelmää on havainnollistettu kuvassa 2.4.

Koska satelliitin lähettämä radiosignaali kulkee valonnopeudella, pystyy vastaanottaja signaalin lähettämisajan perusteella laskemaan oman etäisyytensä kyseiseen satelliittiin. Eli

$$r = c\,(T_u - T_s) = c\Delta t, \tag{2.1}$$

missä r on satelliitin ja käyttäjän välinen etäisyys, c on valonnopeus, T_s on ajanhetki, jolla satelliitti lähettää signaalin ja T_u on aika, jolloin vastaanottaja saa viestin.

Koska vastaanottajan ja satelliitin kellot eivät ole samassa ajassa, täytyy kellojen erosta johtuva erotus t_u ottaa myös huomioon. Tämän vuoksi paikannusta ei voida

tehdä todellisten etäisyyksien perusteella, vaan vastaanottajan ja satelliitin välisen *pseudoetäisyyden* perusteella. Satelliitin ja vastaanottajan väliseksi pseudoetäisyydeksi ρ_i saadaan

$$\rho_i = r + c(t_u - \delta t_s), \tag{2.2}$$

missä t_u on vastaanottajan ja satelliittijärjestelmän välinen kellopoikkeama ja δt on satelliitin atomikellon ja satelliittijärjestelmän välinen kellovirhe. Satelliittien sisäisen kellovirheen seuranta on osa satelliittipaikannusjärjestelmän ylläpitoa ja GPS-satelliitit välittävät sen navigointiviestiensä mukana [27]. Koska satelliittien kellovirhe on merkittävästi vastaanottimen kellovirhettä pienempi, voidaan δt_s jättää huomioimatta.

Yhtälön etäisyys r voidaan esittää käyttäjän ja satelliitin ECEF-koordinaattien avulla (engl. Earth-Centered, Earth-Fixed). Tällöin pseudoetäisyyden yhtälöksi saadaan

$$\rho_i = \sqrt{(x_i - x_u)^2 + (y_i - y_u)^2 + (z_i - z_u)^2} + ct_u, \tag{2.3}$$

missä x_i, y_i ja z_i ovat satelliitin sijainti ja x_u, y_u ja z_u ovat vastaanottajan sijainti.

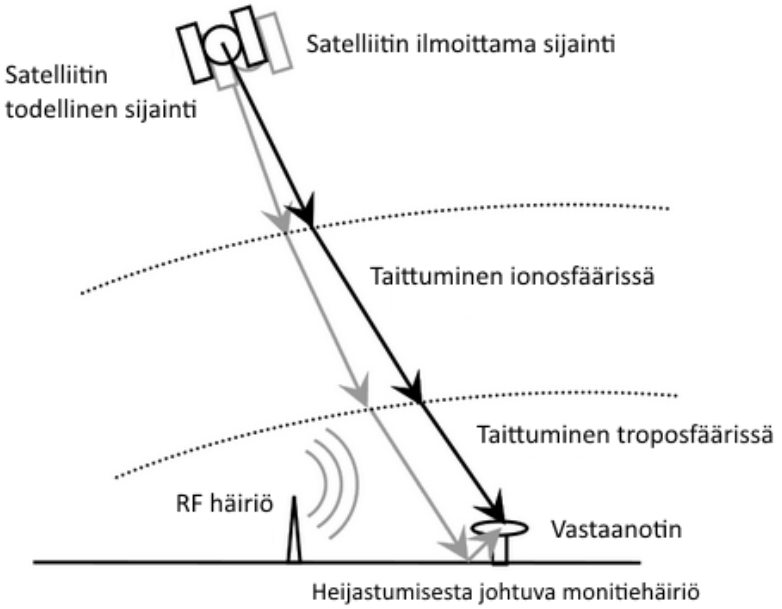
Yhtälössä on neljä tuntematonta muuttujaa: käyttäjän sijainnin koordinaatit x_u, y_u ja z_u, sekä vastaanottimen kellovirhe t_u. Tämän vuoksi yhtälön ratkaisemiseksi tarvitaan vähintään neljän eri satelliitin sijainnit ja pseudoetäisyys, jolloin vastaanottimen lopullinen sijainti on ratkaistavissa yhtälöryhmästä

$$\begin{aligned}
\rho_1 &= \sqrt{(x_1 - x_u)^2 + (y_1 - y_u)^2 + (z_1 - z_u)^2} + ct_u \\
\rho_2 &= \sqrt{(x_2 - x_u)^2 + (y_2 - y_u)^2 + (z_2 - z_u)^2} + ct_u \\
\rho_3 &= \sqrt{(x_3 - x_u)^2 + (y_3 - y_u)^2 + (z_3 - z_u)^2} + ct_u \\
\rho_4 &= \sqrt{(x_4 - x_u)^2 + (y_4 - y_u)^2 + (z_4 - z_u)^2} + ct_u.
\end{aligned} \tag{2.4}$$

Yhtälöryhmä voidaan ratkaista monella eri tavalla, kuten esimerkiksi Taylorin sarjalla tai Kalman-suodattimen avulla [27].

2.2.2 Satelliittipaikannuksen häiriöt

Ideaalioloissa halpakin GNSS-vastaanotin pystyy määrittämään sijaintinsa muutaman metrin tarkkuudella. Valitettavasti satelliittien signaalit ovat herkkiä erilaisille häiriötekijöille [23] [27]. Tässä luvussa esitellään yleisimpiä satelliittipaikannukseen liittyviä häiriötekijöitä.



Kuva 2.5 *Satelliittin virhelähteitä. Kuva [23, Fig. 8.13]*

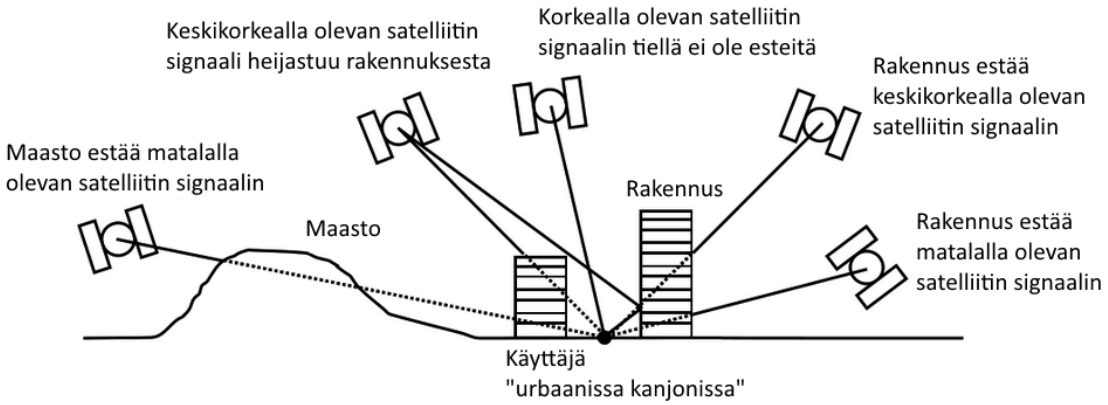
.

Ilmakehän vaikutus

Satelliitin asema vastaanottajasta katsottuna vaikuttaa merkittävästi signaaliin kuuluvuuteen. Mitä matalammalla horisontissa satelliitti on, sitä pidemmän matkan signaali joutuu etenemään ilmakehässä. Valon tapaan myös satelliitin lähettämä signaali taittuu maapallon ilmakehässä, jonka vuoksi satelliitin lähettämä signaali ei kulje suorinta tietä vastaanottajalle (kuva 2.5). Signaalin hidastumisen vuoksi vastaanotin voi luulla olevansa kauempana satelliitista, kuin se todellisuudessa on [23].

Ilmakehästä johtuvien häiriöiden korjaamiseksi on kehitetty erillisiä satelliittipohjaisia parannusjärjestelmiä (engl. Satellite Based Augmentation System, SBAS). Järjestelmät ovat alueellisia ja niiden tarjoamat korjaukset pätevät vain niiden tomintaalueen ilmakehässä [27]. Esimerkiksi GPS-järjestelmän toimintaa parantavia apujärjestelmiä ovat muun muassa:

- Eurooppalainen EGNOS (European Geostationary Navigation Overlay Service)
- Yhdysvaltalainen WAAS (Wide Area Augmentation System)



Kuva 2.6 *Käyttäjä urbaanissa kanjonissa. Käyttäjän GPS vastaanottimen on vaikeaa määrittää käyttäjän sijaintia, sillä vain yhden satelliitin signaalin tiellä ei ole esteitä. Kuva [23, Fig. 8.14]*

- Japanilainen MSAS (Multi-functional Satellite Augmentation System)
- Intialainen GAGAN (GPS Aided Geo Augmented Navigation)

Fyysiset esteet ja heijastukset

Fyysiset esteet, kuten puut, vuoret, rakennukset ja ajoneuvot voivat haitata tai jopa estää satelliittien lähettämien signaalien etenemisen. Varsinkin tiheästi rakennetuissa kaupungeissa rakennukset voivat muodostaa niin sanotun urbaanin kanjonin (engl. urban canyon), jossa rakennukset joko estävät satelliittisignaalit kokonaan tai signaali heijastuu rakennuksista vastaanottimeen [23]. Esimerkiksi kuvan 2.6 tapauksessa neljästä käyttäjää lähimmästä satellitista vain kahden signaali kuuluu ja vain yhden satelliitin signaalin tiellä ei ole esteitä.

Heijastukset ovat haastavia, sillä ne voivat aiheuttaa monitiehäiriöitä (engl. multipath). Monitiehäiriössä vastaanotin saa satelliitin lähettämän signaalin useasta eri lähteestä. Signaali voi saapua vastaanottimelle ensimmäisen kerran vaimentuneena sen suorinta reittiä pitkin ja vähän myöhemmin toisen kerran esimerkiksi maasta, rakennuksesta tai ajoneuvosta heijastuneena. Esimerkiksi kuvassa 2.5 satelliitin lähettämä signaali heijastuu maasta vastaanottimelle, aiheuttaen monitiehäiriön. Monitiehäiriö voi huomattavasti hankaloittaa signaalin vastaanottamista ja tulkitsemista ja siten merkittävästi heikentää saadun pseudoetäisyyden tarkkuutta. [27]

Muita virhelähteitä

Edellä mainittujen virhelähteiden lisäksi satelliitin signaalia voivat häiritä vastaanottimen lähellä olevat muut laitteet, joiden käyttämät radiotaajuudet ovat lähellä satelliittien käyttämiä taajuuksia. On myös mahdollista, että signaalia yritetään tarkoituksella häiritä: vastaanotinta voidaan joko estää vastaanottamasta signaalia (engl. jamming) tai sille voidaan lähettää valheellista signaalia, jolla yritetään saada vastaanotin luulemaan olevansa eri paikassa, kuin missä se oikeasti on (engl. spoofing). [27]

3. KOMPONENTTIEN VALINTA

Järjestelmän suunnittelu aloitetaan järjestelmän käyttötarkoituksen ja yleisten vaatimusten määrittelemisestä. Tämän jälkeen vertaillaan erilaisia laitteita, joista järjestelmä voitaisiin kasata, sekä perustellaan tehdyt valinnat.

3.1 Yleiset vaatimukset

Suunniteltavan järjestelmän pääasiallinen tarkoitus on toimia tarkkana referenssi- ja kalibrointijärjestelmänä halvemmille kuluttajalaatuisille inertiamittausyksiköille. Lisäksi inertiamittausyksikön haluttiin olevan riittävän tarkka, jotta sillä voidaan laskea napapohjoisen suunta maapallon pyörimisliikkeen perusteella. Tämän vuoksi järjestelmään haluttiin vähintään taktista-laatua oleva inertiamittausyksikkö.

Koska järjestelmää tullaan käyttämään ajoneuvoissa, järjestelmään haluttiin myös tarkka satelliittipaikannin. Jotta GPS-järjestelmää käyttävä paikannin olisi mahdollisimman tarkka, sen täytyy pystyä hyödyntämään joko SBAS-järjestelmien tarjoamia parannuksia tai käyttämään hyväksi differentiaalista paikannusta.

Koska järjestelmää tullaan siirtelemään mittauskohteesta toiseen, sen siirtämisen, asentamisen ja purkamisen pitää olla helppoa. Tästä seuraa, että järjestelmän täytyy olla kevyt ja sen pitää mahtua pieneen tilaan. Koon lisäksi järjestelmän virrankulutuksen pitää olla riittävän matala, jotta se toimisi akun varassa vähintään tunnin ajan. Koko- ja painovaatimukset rajoittavat akun valintaa, joten esimeriksi painavat lyijyakut ovat poissuljettu vaihtoehto.

Järjestelmästä täytyy löytyä paikallista tallennustilaa tunnin mittausajanjaksoa vastaava määrä. Tallennustilan tarvittava määrä riippuu valituista antureista ja näiden dataformaatista, minkä vuoksi tallennuskapasiteetille ei voi etukäteen antaa yksiselitteistä suuruutta.

Järjestelmän toteutuksen yhteydessä havaittiin tarve mittauksien etäkäynnistyksel-

(a) Matrix-504 [4] (b) Raspberry Pi 2 Model B [38]

Kuva 3.1 *Vertaillut pienoistietokoneet.*

le. Tämän vuoksi laitteistolle asetettiin toissijainen vaatimus mittauksien käynnistämiselle ja lopettamiselle etäältä esimerkiksi Bluetooth-yhteyden avulla.

3.2 Järjestelmän osat

Edellä esiteltyjen vaatimusten perusteella selvitettiin, minkälaisista osista järjestelmä voitaisiin rakentaa. Seuraavissa aliluvuissa on esitelty erilaisia vaihtoehtoja järjestelmän ohjausyksiköksi, inertiamittausyksiköksi ja GNSS-vastaanottimeksi, sekä perustellaan näiden valinnat.

3.2.1 Ohjausyksikkö

Kuten luvussa 3.1 todettiin, laitteistosta haluttiin pienikokoinen, jotta sen siirtäminen, asentaminen ja purkaminen olisi mahdollisimman helppoa ja yksinkertaista. Tämän vuoksi pääyksiköksi haluttiin yhdelle piirilevylle mahtuva pienoistietokone, josta löytyy monipuolisesti liitäntöjä. Työssä päädyttiin vertaamaan keskenään Matrix-504 [4] ja Raspberry Pi 2 Model B [38] -pienoistietokoneita. Kuvat laitteista löytyvät kuvasta 3.1 ja niiden ominaisuuksia on vertailtu taulukossa 3.1.

Taulukko 3.1 *Pienoistietokoneiden ominaisuuksia. [4] [38]*

	Matrix-504	**Raspberry Pi 2**
CPU	ATMEL 400MHZ AT91SAM9G20	900MHz quad-core ARM Cortex-A7
Muisti	64 MB SDRAM 256 MB NAND Flash	1 GB LPDDR2
DataFlash	2 MB	
Sarjaportit	4 × RJ45	Ei
- RS232	4	
- RS422	1	
- RS-485	4	
Muut liitännät		
Ethernet	Kyllä	Kyllä
GPIO	5 kpl	27 kpl
USB 2.0	2 kpl	4 kpl
SD-korttipaikka	Kyllä	Kyllä (SD-kortti sisältää käyttöjärjestelmän)
Muut		DSI, HDMI, I^2C, MIPI, SPI
Virrankulutus		
Jännite	12 VDC	5 V
Virta	250 mA	2 A
Teho	$<$ 3 W	$<$ 10 W
Ohjelmisto		
Käyttöjärjestelmä	Linux, kernel 2.6.29	Useita vaihtoehtoja
Hinta	300-500 €	$>$ 35 €

Matrix-504

Matrix-504 on taiwanilaisen Artila -yrityksen suunnittelema ARM-9 -pohjainen sulautettu tietokone, joka on tarkoitettu internettiin kytkettyjen esineiden ja laitteiden (engl. Internet of Things, IoT) yhdyskäytäväksi, sekä IoT-sovellusten laskentaalustaksi. Matrixin datalehdeltä [4] ja taulukosta 3.1 nähdään, että laite voisi hyvinkin sopia mittauslaitteiston laskentayksiköksi. Matrix-504:sta löytyy monipuolisesti eri liitäntöjä, joten se ei rajoita vaihtoehtoisia mittalaitteita sopivan liitännän puuttumisen vuoksi. Lisäksi SD-korttipaikka mahdollistaa mittausdatan helpon tallenta-

misen ja siirtämisen eri koneiden välillä ilman, että laitteisto vaatisi internetyhteyden tai ulkoisen kovalevyn. Toisaalta, jos laitteisto haluttaisiin kytkeä internettiin, Matrixista löytyy ethernet-portti tätä varten. [4]

Matrix-504 laitteessa on valmiiksi asennettuna Linux 2.6.29 -ytimellä (engl. kernel) toimiva käyttöjärjestelmä. Lisäksi laitteelle voidaan kehittää ohjelmistoa vapaasti saatavissa olevilla GNU C/C++ -kehitystyökaluilla [19] [4]. Toisin sanoen ohjelmoitavuudeltaan laite vaikuttaa tyypilliseltä ARM-pohjaiselta GNU/Linuxilta, joten laitteen ohjelmoinnissa voi hyvinkin saada tukea Artilan lisäksi esimerkiksi Linux-kehittäjäyhteisöltä. Tämän lisäksi laitteesta löytyy valmiiksi konfiguroidut SSH- ja FTP-palvelimet, joten laitteella voidaan aloittaa työskentely hyvinkin nopeasti.

Matrix-504:lla on verrattain pieni noin 3 W tehonkulutus ja valmistajan mukaan se kestää myös ankariakin teollisia käyttöympäristöjä [4]. Nämä ominaisuudet tekevät Matrixista mielenkiintoisen alustan suunniteltavalle laitteistolle, sillä järjestelmää tullaan käyttämään myös työmaaympäristöissä. Valitettavasti nämä ominaisuudet näkyvät myös laitteen hinnassa, sillä laitteen hinta voi olla 300–500 euroa riippuen jälleenmyyjästä.

Raspberry Pi 2

Raspberry Pi on englantilaisen Raspberry Pi -säätiön kehittämä tietokoneperhe, jonka tavoitteena on tarjota edullinen alusta ohjelmoinnin opettelua ja opettamista varten. Edullisen hintansa vuoksi Raspberry Pi:sta tuli todella suosittu opettajien lisäksi alan harrastajien keskuudessa. Opetus- ja harrastuskäytön lisäksi Raspberry Pi -koneita käyttävät nykyään myös tutkimuslaitokset ja yritykset[21]. Suunniteltavaa laitteistoa varten tarkasteltiin Raspberry Pi 2 -pienoistietokonetta.

Matrix-504 poiketen Raspberry Pi:ssä ei ole yhtään sarjaportti liitäntää, mutta GPIO (engl. General Purpose Input Output) -pinnien ja USB-porttiensa ansiosta Raspberry Pi:hin voidaan silti liittää monipuolisesti erilaisia laitteita. Tämän lisäksi Raspberry Pi:stä löytyy myös kaksi erilaista näyttöliitäntää, siinä missä Matrix-504:ää ei saa kytkettyä näyttöön lainkaan.

Matrix-504:n tavoin Raspberry Pi:llä voidaan ajaa Linux-pohjaista käyttöjärjestelmää. Raspberry Pi säätiön virallinen suositus on Raspbian käyttöjärjestelmä [43], joka on Raspberry Pi alustalle räätälöity versio Debian -käyttöjärjestelmästä [13].

Virallisen Raspbianin lisäksi myös muista suosituista Linux-jakeluista, kuten esimerkiksi Arch Linuxista, Fedorasta ja Ubuntusta on tehty Raspberry Pi:lle sopivat versiot Arch Linux ARM, Pidora ja Ubuntu ARM [2][3][17][40] [47][46]. Linux-jakeluiden lisäksi Raspberry Pi:lle voidaan asentaa myös Microsoftin Windows 10 IoT Core-käyttöjärjestelmä [30].

Käyttöjärjestelmä asennetaan ja käynnistetään Raspberry Pi:n SD-kortilta, joten Linux-jakeluiden osalta SD-kortille joudutaan alustamaan Linux-tiedostojärjestelmä. Tämä monimutkaistaa SD-kortin käyttöä, sillä Windows-käyttöjärjestelmät eivät voi suoraan lukea Raspberry Pi:n tiedostojärjestelmään tallennettua tiedostoa ilman erillistä apuohjelmaa. Toinen vaihtoehto on tallentaa mittausdata SD-kortin sijaan ulkoiselle tiedontallennusvälineelle, kuten esimerkiksi USB-muistille. Tämä tekee Raspberry Pi:sta hieman hankalamman käyttää, kuin Matrix-504, jonka SD-kortille tallennetun datan voi siirtää vaivatta koneesta toiseen.

Raspberry Pi 2 käyttää ARM Cortex-A7 -moniydinprosessoria, jonka laskentateho on suurempi, kuin Matrix-504:n käyttämän AT91SAM9G20. Toisaalta siinä missä Matrix-504 sähköteho on 3 W, Raspberry Pi:n sähköteho on 10 W. Raspberry Pi kuluttaa siis noin 3 kertaa enemmän energiaa kuin Matrix-504.

Ohjausyksikön valinta

Matrix-504:stä löytyy enemmän sarjaporttiliitäntöjä, kuin Raspberry Pi:sta, sen virrankulutus on selvästi pienempi ja mittausdatan tallentaminen ja lukeminen olisi sillä helpompaa, kuin Raspberry Pi:lla. Raspberry Pi vuorostaan peittoaa Matrix-504:n hinnassa, USB porttien määrässä, sekä monipuolisimmissa liitännöissä. Koska Raspberry Pi on erittäin suosittu harrastajien keskuudessa, joten sille todennäköisesti löytyy myös enemmän tukea ongelmatilanteisiin, kuin Matrix-504:lle. Tämän tarkastelun perusteella järjestelmän ohjausyksiköksi valittiin Raspberry Pi 2.

3.2.2 Inertiamittausyksikkö

Järjestelmän yleisissä vaatimuksissa todettiin, että järjestelmää käytetään halvempien ja vähemmän tarkempien inertiamittausyksiköiden referenssi ja kalibrointilaitteena, sekä todellisen pohjoisen löytämiseen. Jotta näihin tavoitteisiin päästäisiin, pitää järjestelmän inertiamittausyksikön olla vähintään neljä kertaa tarkempi, kuin

(a) iMar iFOG-IMU-1-A [25]

(b) KVH 1750 IMU [29]

Kuva 3.2 *Vertaillut inertiamittausyksiköt*

maan pyörimisnopeus $15°/h$ [10]. Valitettavasti näin tarkat inertiamittausyksiköt, voivat maksaa useita kymmeniä tuhansia tai satoja tuhansia euroja [23]. Tämän vuoksi inertiamittausyksikön hinta on erittäin oleellinen kriteeri valittavalle referenssilaitteelle. Lisäksi inertiamittausyksikön fyysinen koko on myös otettava huomioon, eikä se saa olla liian painava tai isokokoinen.

Työtä varten vertailtiin kahta kuituoptista inertiamittausyksikköä. Ensimmäinen oli saksalaisen iMar:n valmistama iFOG-IMU-1-A [25] ja toinen yhdysvaltaisen KVH Industries:n valmistama KVH 1750 IMU [29]. Kuvat mittausyksiköistä löytyvät kuvasta 3.2 ja näiden tärkeimmät ominaisuudet löytyvät taulukosta 3.2.

iMar iFog-IMU-1-A

Taulukosta 3.2 löytyvien ominaisuuksien perusteella iFog voisi sopia järjestelmän inertiamittausyksiköksi. Se on riittävän tarkka sekä halvempien inertiamittausyksiköiden kalibrointia varten, että maan pyörimisen havainnointia varten. Se myös painaa alle kilon ja on sopivan kokoinen siirreltäväksi mittauspaikalta toiselle. iFOG-IMU-1-A:n tehonkulutus on verrattain iso verrattuna KVH 1750:n tehonkulutukseen.

iFOG-IMU-1-A:sta löytyy MIL–DTL–83513 standardin [14] mukainen 25-pinninen micro–D -liitäntä ja se käyttää RS422-protokollaa. Väylän baudinopeus 909.1 kBd

Taulukko 3.2 *Inertiamittausyksiköiden ominaisuuksia [25] [29]*

	iFog-IMU-1-A	**KVH 1750 IMU**
Kiihtyvyysanturi		
Mittausalue	$\pm$ 5 g	$\pm$2 g
Vinouma	2 mg	1.5 mg
Satunnaisliike	$< 50\ \mu g/\sqrt{Hz}$	$< 24\ \mu g/\sqrt{Hz}$
Gyroskooppi		
Mittausalue	$\pm$ 450 $^\circ/s$	$\pm$ 490 $^\circ/s$
Vinouma	$< 0.75\ deg/hr$	$< 0.1\ deg/s$
Satunnaisliike	$< 0.15^\circ/\sqrt{h}$	$\leq 0.012\ ^\circ/\sqrt{h}$
Dataliikenne		
Liitin	Micro SubD 25 pin	Micro-D 15 pin
Sarjaliikenne	RS422	RS422
Tiedonsiirtonopeus	909.1 kBd	901.6 kBd
Datavauhti	0-1000 Hz	1-1000 Hz
Virrankulutus		
Jännite	15 V, +5 VDC	9-36 VDC
Teho	9-12 W, max 18 W	5 W, max 8 W
Koko ja paino		
Dimensiot	110 $\times$ 83 $\times$ 77 mm	88.9 $\times$ 73.7 mm
Paino	950 g	700 g

ja datanopeus on maksimissaan 1000 Hz. Tämä on hankala yhdistelmä ominaisuuksia, sillä micro–D on sotilasstandardi, eikä sitä juuri käytetä kaupallisissa sovelluksissa. Tämän lisäksi Raspberry Pi:sta ei löydy suoraan tukea RS422-protokollalle, joten järjestelmään jouduttaisiin hankkimaan erillinen RS422–USB -sovitin. Tältä kannalta Matrix-504 olisi sopinut sovellukseen paremmin, kuin Raspberry Pi valmiin RS422 porttinsa vuoksi. Toisaalta Matrix-504 olisi kuitenkin vaatinut Micro-D–RJ45 -sovittimen.

KVH 1750 IMU

KVH 1750 IMU on KVH Industries:n valmistama inertiamittausyksikkö, jonka ominaisuudet ovat samaa luokkaa iMarin iFog-mittausyksikön kanssa. Taulukon 3.2 tietojen perusteella se on iFogia hieman tarkempi ja sen kiihtyvyysantureiden mittausalue on hieman iFogia pienempi. KVH 1750:n antureiden vinouma ja satunnais-

liike ovat iFogia pienemmät. Gyroskoopin mittausalue vuorostaan on isompi ja sen vinoumasta ja satunnaisliikkeestä seuraavat virheet ovat pienemmät.

Dataliikenteen osalta molemmat inertiamittausyksiköt käyttävät Micro–D liitäntää, mutta KVH 1750 käyttää 25 pinnisen liittimen sijaan 15-pinnistä liitäntää. Molemmat käyttävät RS422-protokollaa ja kummankin maksimi datavauhti on 1000 Hz. KVH 1750:n maksimi siirtonopeus on 921.6 kBd, joka on hieman enemmän, kuin iMarin anturissa.

Toiminnallisten vaatimusten lisäksi inertiamittausyksiköltä edellytettiin pientä kokoa, painoa ja virran kulutusta. Näissä suhteissa KVH 1750 on iMarin mittausyksikköä parempi joka suhteessa. KVH 1750 on iMarin mittausyksikköä pienempi ja kevyempi ja se kuluttaa vähemmän virtaa, kuin iMarin mittausyksikkö. Lisäksi työtä varten esitettyjen tarjouspyyntöjen perusteella KVH 1750 oli selvästi halvempi, kuin iMarin iFog-IMU-1-A. Tämän vertaillun perusteella järjestelmän inertiamittausyksiköksi valittiin KVH 1750 IMU.

3.2.3 Satelliittivastaanotin

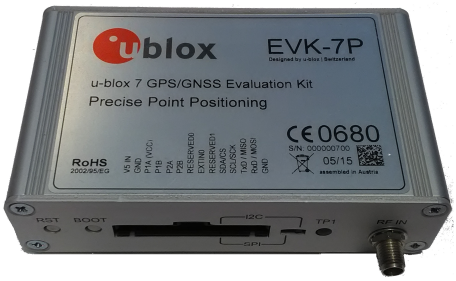
Järjestelmän vaatimuksissa todettiin, että järjestelmän satelliittivastaanottimen tulisi olla tarkka ja helposti siirrettävissä paikasta toiseen. Lisäksi paikantimelta edellytettiin tavallista satelliittipaikanninta parempaa tarkkuutta.

Tietotekniikan laitoksen Navigation Group:lla oli jo entuudestaan kanadalaisen NovAtelin *DL-4 plus* GPS-vastaanottimia [32], mutta näiden iän vuoksi työtä varten selvitettiin myös uudemman vastaanottimen ominaisuuksia. Vaihtoehtoiseksi vastaanottimeksi valittiin sveitsiläisen u-blox yrityksen kehittämä *EVK-7P* [45]. Kuvassa 3.3 on työssä vertaillut vastaanottimet ja näiden ominaisuuksia on vertailtu taulukossa 3.3.

NovAtel DL-4 Plus

DL-4 plus on kanadalaisen NovAtel yrityksen kehittämä tehokas GPS-vastaanotin, joka pystyy itsenäisesti tallentamaan mittausdataa muistikortille [32]. Laite on suunniteltu toimimaan itsenäisenä GPS-vastaanottimena, eikä se siis välttämättä tarvitse ulkoista ohjausyksikköä. Toisaalta vastaanotin voidaan myös kytkeä Raspberry

(a) NovAtel DL-4 Plus



(b) u-blox EVK-7P

Kuva 3.3 *Vertaillut satelliittivastaanottimet.*

Taulukko 3.3 *Satelliittipaikantimien ominaisuuksia*

	DL-4 Plus	**EVK-7P**
GPS:n ominaisuudet		
Paikannustarkkuus	1.5 m, RT-20 $<$ 20 cm	2.5 m, PPP $<$ 1.0 m
Aika 1. ratkaisuun	30-50 s	30 s
Aikatarkkuus	20 ns RMS	30 ns RMS
Nopeustarkkuus	0.03 m/s RMS	0.1 m/s
Liitännät		
Data	4 x DB9 RS232	I^2C, SPI, USB, RS232
Antenni	TNC female	SMA female, RF
	+5 VDC, max 100mA	3.3V, max 30mA
Muistikortti	Kyllä	Ei
Virrankulutus		
Jännite	9–18 VDC	3.6–5 V
Teho	3.5 W	Ei saatavilla
Koko ja paino		
Dimensiot	185 $\times$ 154 $\times$ 71 mm	105 $\times$ 64 $\times$ 26 mm
Paino	1.2 kg	Ei saatavilla

Pi:hin RS232-USB sovittimen avulla, jolloin Raspberry Pi pystyy ohjaamaan ja lukemaan vastaanotinta.

DL-4 Plus:sta löytyy valmiiksi aikaleimaamistoiminnallisuus, jonka avulla se pystyy aikaleimaamaan sen lävitse ohjatun sarjadatan. Vastaanotinta voidaan valmistajan mukaan käyttää *realiaikaiseen kinemaattiseen mittaamiseen* (RTK), eli sen tarkkuus voi olla sopivissa olosuhteissa jopa senttimetrien luokkaa.

DL-4 Plus -vastaanotin on melko kookas ja painava, jonka lisäksi se vaatii oman akkunsa. Tämä yhdistelmä tekee siitä vaikeasti siirrettävän ja siten rajoittaa hieman sen käyttöä. Lisäksi, jotta DL-4 Plus pääsisi parhaimpaan tarkkuuteen, pitäisi mittauskohteen lähettyville pystyttää erillinen tukiasema RTK-mittauksia varten.

u-blox EVK-7P

EVK-7P on sveitsiläisen u-blox -yrityksen valmistama PPP (engl. Precise Point Positioning) GPS/GLONASS vastaanotin [45]. EVK-7P voidaan liittää Raspberry Pi:n RS232:n lisäksi myös USB, I^2C tai SPI liitännällä ja se kuluttaa virtaa vain murto-osan siitä mitä DL-4 Plus. EVK-7P voi ottaa kaiken tarvitsemansa virtansa USB-väylän kautta, joten se ei tarvitse ulkoista virtalähdettä.

Suorituskyvyn puolesta EVK-7P on hieman erilainen DL-4 Plus:n verrattuna. EVK-7P saa DL-4 Plussaa nopeammin ensimmäisen sijaintiratkaisunsa (*Time To First Fix, TTFF*). Lisäksi sen normaali paikannustarkkuus on samalla tasolla, kuin DL-4 Plussassa, vaikka sen käyttämä PPP-algoritmi ei olekaan yhtä tarkka, kuin DL-4 Plussan RTK-tekniikka. EVK-7P:n PPP-algoritmi käyttää hyväkseen SBAS-satelliiteista saatavaa ilmakehätietoa, jonka avulla se pystyy korjaamaan ilmakehästä johtuvia virheitä.

u-blox:n EVK-7P huomattiin kilpailukykyiseksi kandidaatiksi järjestelmän satelliittivastaanottimeksi. Sen pieni koko ja virrankulutus ovat selviä etuja NovAtelin DL-4 Plussaan verrattuna. EVK-7P ei kuitenkaan ole aivan yhtä tarkka, kuin DL-4 Plus, joten EVK-7P ei tule täysin korvaamaan jo olemassa olevia DL-4 Plus vastaanottimia. Toisaalta DL-4 Plus havaittiin hyvin kömpelöksi, jonka vuoksi työssä päädyttiin hankkimaan u-bloxin EVK-7P esimerkiksi kaupungissa tapahtuvia mittauksia varten.

(a) HTD22110A (b) CS-LP5004C35RT

Kuva 3.4 *Järjestelmän akut*

3.2.4 Muut laitteet

Edellä tehtyjen laitevalintojen perusteella laitteistolle valittiin sopivat akut. Työssä päädyttiin alustavasti hankkimaan erilliset akut sekä Raspberry Pi:lle, että KVH 1750:lle. EVK-7P saa virtansa Raspberry Pi:n USB portin kautta, eikä se näin ollen tarvitse erillistä virtalähdettä. Kuvassa 3.4 on järjestelmää varten hankitut akut.

Raspberry Pi:n akuksi hankittiin 12000 mAh HTD22110A akku [12], josta löytyy suoraan USB ulostulo Raspberry Pi:ta varten. Akku on riittävän hyvä, jotta sillä voidaan käyttää Raspberry Pi:ta ja siihen liitettyjä laitteita yli kolmen tunnin ajan. KVH:ta varten hankittiin Cameron Sino:n CS-LP5004C35RT akku, jonka jännite on 14.8V ja kapasiteetti on 5000 mAh. Teoriassa akussa riittää virtaa KVH:lle yli 9 tunnin ajaksi.

Cameron Sino-akku on riittävän tehokas, jotta sillä voitaisiin hoitaa sekä inertiamittausyksikön että Raspberry Pi:n virransyöttö. Varteenotettava jatkokehitysidea olisikin asentaa muuntaja Cameron Sinon ja Raspberry Pi:n väliin, jolloin järjestelmä vaatisi vain yhden akun. Tämä kuitenkin edellyttäisi sopivan jännitteen alentimen asentamisen Raspberry Pi:n ja Cameron Sino-akun väliin, sillä Raspberry Pi:n käyttöjännite on vain 5 V. Lisäksi alennin pitäisi kytkeä Raspberry Pi:n micro-USB porttiin, sillä Raspberry Pi:n GPIO pinnejä ei ole suojattu regulaattorilla [38]. Järjestelmään hankittiin myös erillinen Bluetooth-sovitin mittauksien etäkäynnistystä varten.

4. LAITTEISTOKUVAUS

Tässä luvussa käydään tarkemmin lävitse luvussa 3 valittujen laitteiden ominaisuuksia, sekä kuvataan laitteiden kytkennät ja konfigurointi. Kuvassa 4.1 on valokuva koko laitteistosta ja kuvassa 4.2 pelkistetty kuva laitteiston kytkennöistä.

4.1 Ohjausyksikkö

Raspberry Pi on järjestelmän ydin ja kaikki laitteet KVH 1750:n akkua lukuun ottamatta kytketään siihen kuvan 4.2 mukaisesti. Raspberry Pi saa virtansa HTD22110A-akun 2.4 ampeerin USB-portista, joka kytketään Raspberry Pi:n omaan Micro-USB-porttiin. KVH 1750 IMU ja u-blox EVK-7P kytketään Raspberry Pi:n USB-portteihin.

4.1.1 Käyttöjärjestelmä ja yhdyskäytävät

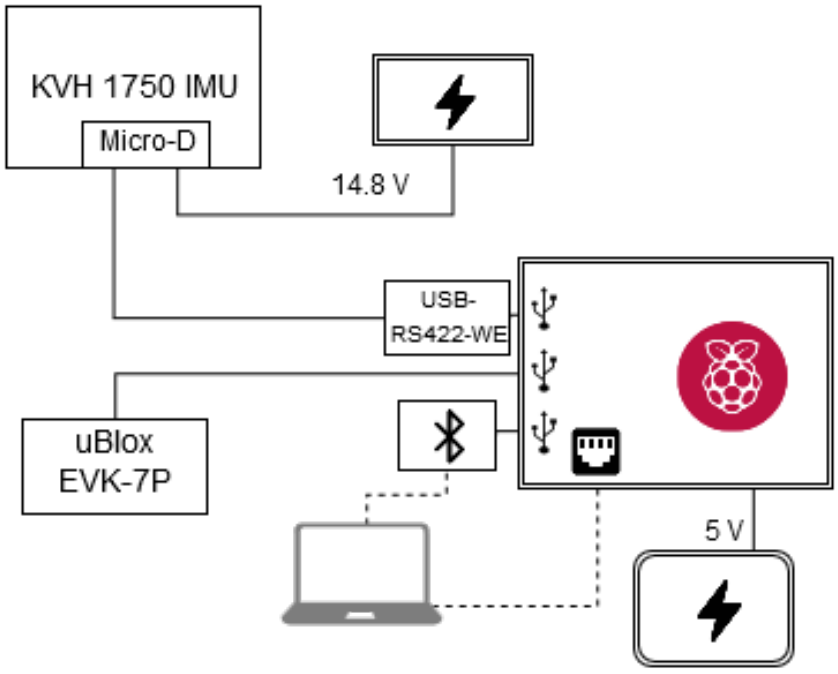
Raspberry Pi:lle asennettiin Raspberry Pi säätiön suosittelema Raspbian käyttöjärjestelmä [43], joka on käytännössä Raspberrylle räätälöity versio Debian Wheezy [13] käyttöjärjestelmästä. Käyttöjärjestelmä oli SD-kortin luonnin jälkeen välittömästi käyttövalmis, eikä sitä tarvinnut erikseen asentaa.

Raspberry Pi:tä ja muuta laitteistoa voidaan ohjata avaamalla Raspberry Pi:lle komentoriviyhteys. Kaksi helpointa tapaa tähän ovat joko kytkeä Raspberry Pi ja päätekone samaan verkkoon ja ottaa Raspberry Pi:lle SSH-yhteys (*Secure Shell*) tai vaihtoehtoisesti kytkeä tietokone Raspberry Pi:n UART-pinneihin sopivan kaapelin avulla ja käyttää komentoriviä sarjaportti-yhteyden kautta.

4.1.2 Bluetoothin konfigurointi

Bluetooth-palvelin ohjelmisto vaatii toimiakseen Linuxin virallisen Bluetooth-toteutuksen Bluez [7] asentamisen sekä Raspberry Pi:n ja sitä käyttävän Bluetooth-laitteen yh-

Kuva 4.1 *Mittauslaitteisto. Taka-alalla Raspberry Pi:n ja KVH 1750:n akut. Etualalla Raspberry Pi, KVH 1750 IMU ja u-blox EVK-7P.*



Kuva 4.2 *Laitteiston kytkennät. EVK-7P ja Bluetooth-sovitin voidaan suoraan kytkeä Raspberry Pi:n, mutta KVH 1750 vaatii erillisen sovittimen näiden väliin. Raspberry Pi:ta voidaan ohjata joko Bluetooth:n, ethernetin tai UART:n kautta. KVH 1750 ja Raspberry Pi käyttävät eri akkuja eri käyttöjännitteiden vuoksi.*

distämisen Bluetooth-laitepariksi. Laitepari voidaan muodostaa listauksen 4.1 komennoilla.

```
# Selvitetaan Bluetooth adapterin MAC-osoite
# ja varmistetaan, etta sovitin toimii oikein
hciconfig
# Tehdaan Raspberry Pi:sta nakyva lahettyvilla
# oleville Bluetooth-laitteille
sudo hciconfig hci0 piscan
# Asetetaan Raspberry Pi hyvaksymaan yhdistamispyynnot.
# Kaskyn parametri 1234 on yhdistamisessa kaytettava PIN-koodi
sudo bluetooth-agent 1234
```

Listaus 4.1 *Bluetooth-laiteparin luominen*

Bluetoothia voidaan tarvittaessa käyttää myös komentoriviyhteyden muodostamiseen. Tämä voidaan tehdä avaamalla Raspberry Pi:hin RFCOMM-protokollan mukainen Bluetooth-portti ja yhdistämällä se Linuxin *agetty*-ohjelmaan [6] [18]. Jotta asiakaslaite voisi löytää portin, se pitää tehdä SDP-protokollalla löydettäväksi (engl. Service Discovery Protocol [15]). Listauksessa 4.2 on luetteloitu Bluetooth-yhdyskäytävän luomiseen vaadittavat komennot.

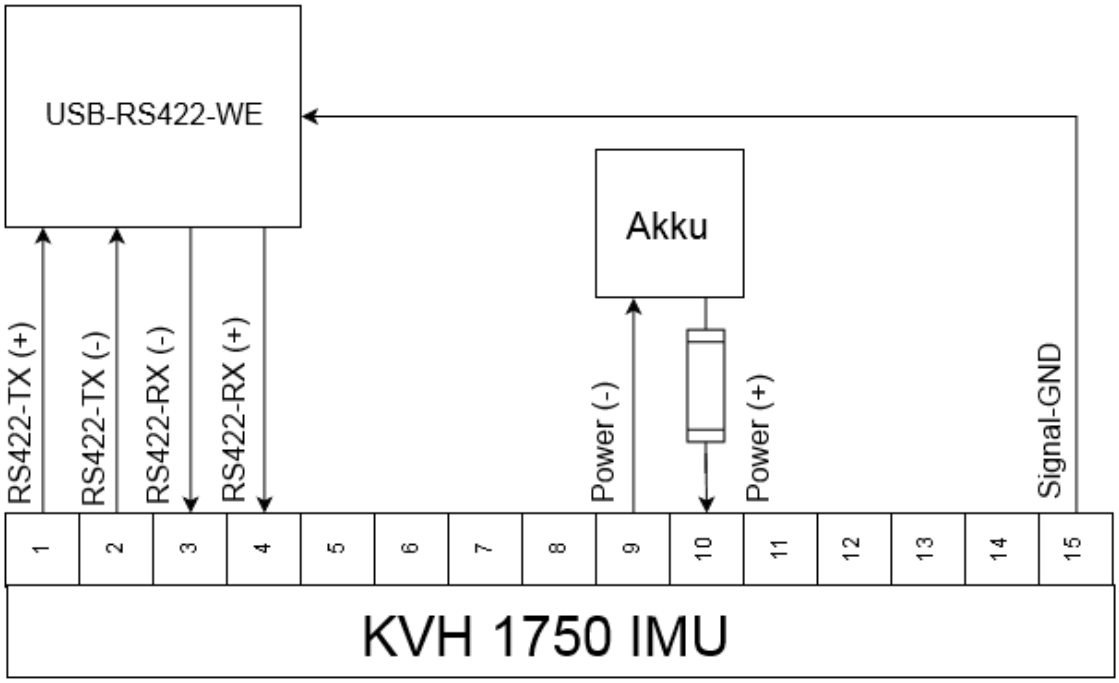
```
# Tehdaan portista 3 loydettava SDP:lla
sdptool add --channel=3 SP
# Luodaan sarjaportti ja yhdistetaan se agettyyn
sudo rfcomm watch /dev/rfcomm0 3 /sbin/agetty rfcomm0 linux
   115200
```

Listaus 4.2 *Bluetooth-yhdyskäytän luominen*

Jotta Bluetooth-komentorivi olisi käytettävissä heti Raspberry Pi:n käynnistyttyä, täytyy edellä mainitut komennot asettaa ajettavaksi käynnistymisen yhteydessä.

4.2 Inertiamittausyksikkö

Järjestelmän inertiamittausyksiköksi valittu KVH 1750 osoittautui järjestelmän monimutkaisemmaksi osaksi sen käyttämän liittimen ja ison käyttöjännitteen vuoksi. Tässä luvussa kuvataan KVH 1750:n kytkentä Raspberry Pi:n ja inertiamittausyksikön käyttämä dataformaatti.

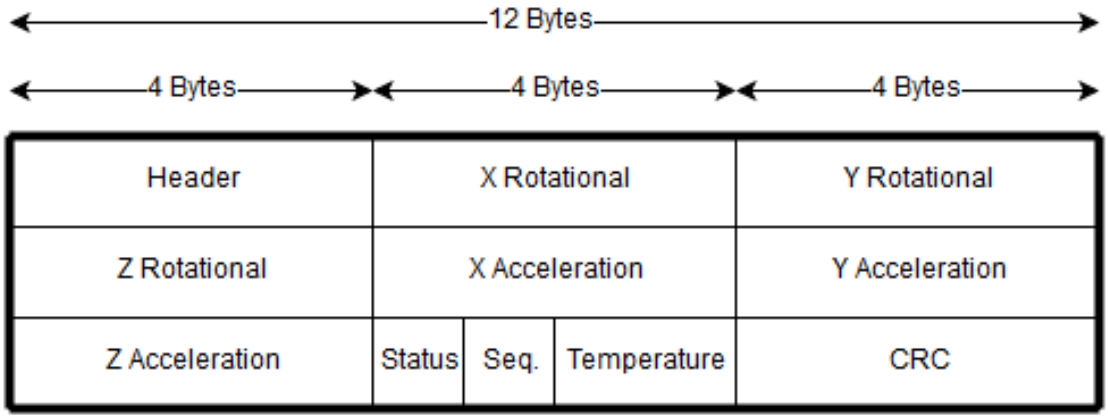


Kuva 4.3 *Inertiamittausyksikön kytkennät. KVH 1750:n datapinnit (1-4), sekä maa ovat yhdistetty suoraan USB-RS422-WE -sovittimeen. Mittausyksikön virtapinnit (9-10) ovat kytketty Cameron Sino akkuun. Akun ja inertiamittausyksikön välissä on sulake suojaamassa mittausyksikköä virtapiikeiltä.*

4.2.1 Kytkennät

KVH 1750:n kytkeminen Raspberry Pi:hin ei ole yksioikoinen operaatio. Ensinnäkin Raspberry Pi 2:stä ei löydy sarjaporttia, joka suoraan tukisi inertiamittausyksikön käyttämää RS422 protokollaa. Tämän vuoksi Raspberry Pi:n ja KVH 1750:n välille laitettiin FTDI:n USB-RS422-WE adapteri [20], jonka avulla mittausyksikön RS422 standardin mukainen liikenne voidaan muuttaa Raspberry Pi:n tukeman USB 2.0 mukaiseksi. USB-RS422-WE adapteri osoittautui hyväksi valinnaksi, sillä sen vihreästä ledistä voidaan nähdä, milloin Raspberry Pi lukee KVH 1750:lta tullutta dataa.

Toinen ongelma oli KVH 1750:n Micro-D liitäntä, jonka kautta dataliikenteen lisäksi KVH 1750:lle syötetään sen käyttöjännite. Inertiamittausyksikköä varten piti siis rakentaa erillinen kaapeli, joka kytkee KVH 1750:n dataliitännät USB-RS422-WE -sovittimeen ja virransyötön Cameron Sino-akkuun. Kytkentään lisättiin sulake suojaamaan mittausyksikköä mahdollisilta virtapiikeiltä.



Kuva 4.4 *[Inertiamittausyksikön viestiformaatti. [29].*

Yksityiskohtainen esitys KVH 1750:n kytkennöistä on esitetty kuvassa 4.3. Kuvassa on mukana myös mittausyksikön ylimääräiset pinnit, joilla voidaan esimerkiksi tarkastella mittauksen paikkansapitävyyttä (pinni 12) tai antaa mittausyksikölle ulkoinen kellosignaali (pinni 11) [29].

4.2.2 Dataliikenne ja -formaatti

Inertiamittausyksikkö aloittaa mittaamisen ja mittausdatan välittämisen heti käynnistymisensä jälkeen, joten sille ei tarvitse välittää erillistä mittauksen aloittamissignaalia. Oletusarvoisesti KVH 1750:n näytteenottotaajuus on 1000 Hz ja väylän siirtonopeus 921.6 kBd. [29]

KVH 1750:n mittausdata välitetään 36 tavun pituisissa binääriviesteissä, joiden rakenne on esitetty kuvassa 4.4. Viestin ensimmäiset 4 tavua muodostavat otsikkotietueen (`0xFE81FF55`), joka ilmaisee uuden viestin alkamista. Seuraavat 12 tavua ovat gyroskooppien mittaamat kiertymisliikkeet x-, y- ja z-akseleiden suhteen ja sitä seuraavat 12 tavua ovat kiihtyvyysantureiden mittaama kiihtyvyys niin ikään kolmen akselin suhteen. Kulmanopeus ja kiihtyvyys ilmaistaan kummatkin kolmella neljän tavun mittaisilla liukuluvuilla; yksi liukuluku kutakin akselia kohden.

Varsinaisen mittausdatan lisäksi viestissä on mukana myös mittaukseen liittyvää metatietoa. Dataviestin 25. tavu ilmaisee, toimivatko kaikki anturit oikein vai toimiiko jokin mittausyksikön antureista virheellisesti. 26. tavu on sekvenssinumero, jonka avulla voidaan havaita, jos dataviestejä on kadonnut mittauksen aikana. Viestin ta-

vut 27 ja 28 ilmaisevat KVH 1750:n sisäisen lämpötilan. Jokaisen viestin lopusta löytyy 4:n tavun mittainen syklinen tarkistusnumero (engl. Cyclic Redundancy Check, CRC), jonka avulla voidaan päätellä, onko viesti tullut alkuperäisessä muodossaan, vai onko siinä mahdollisesti bittivirheitä.

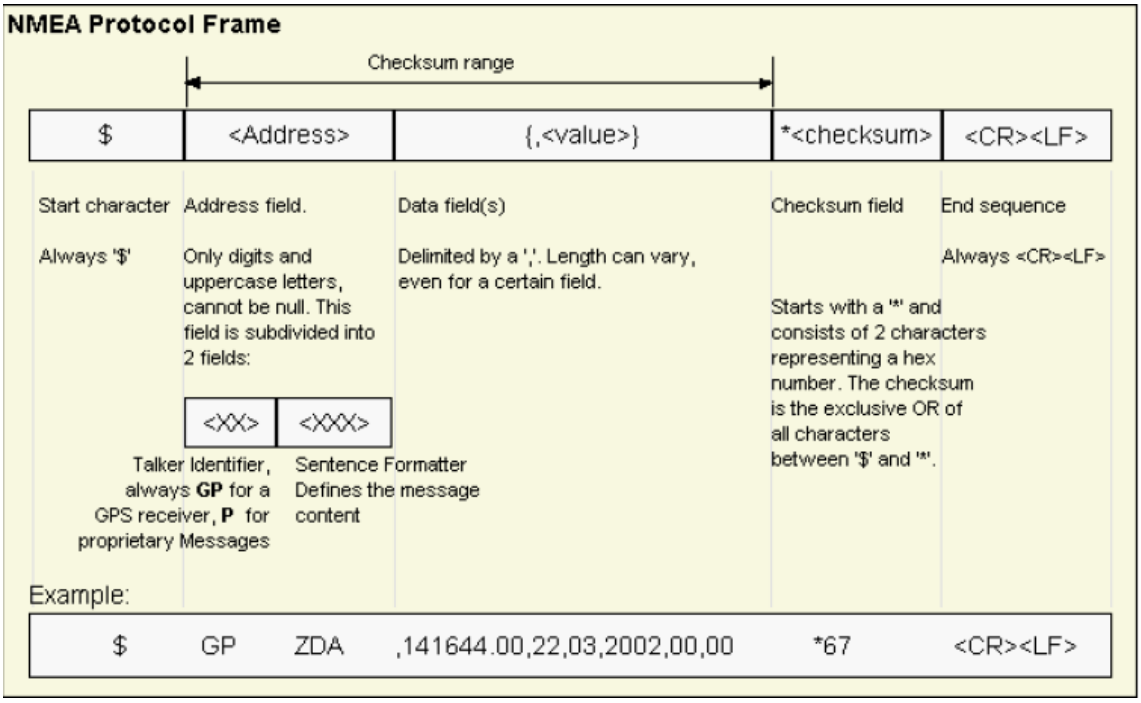
KVH 1750:n toiminnallisuutta voidaan konfiguroida asettamalla se konfiguraatiotilaan. Tällöin laite lopettaa mittaamisen ja jää odottamaan konfiguraatiokäskyjä. Konfiguraatiomoodissa voidaan säätää mm. mittausyksikön käyttämiä datavauhtia ja siirtonopeutta, mittausdatan yksiköitä, sekä käytössä olevia suodatinalgoritmeja.

4.3 Satelliittivastaanotin

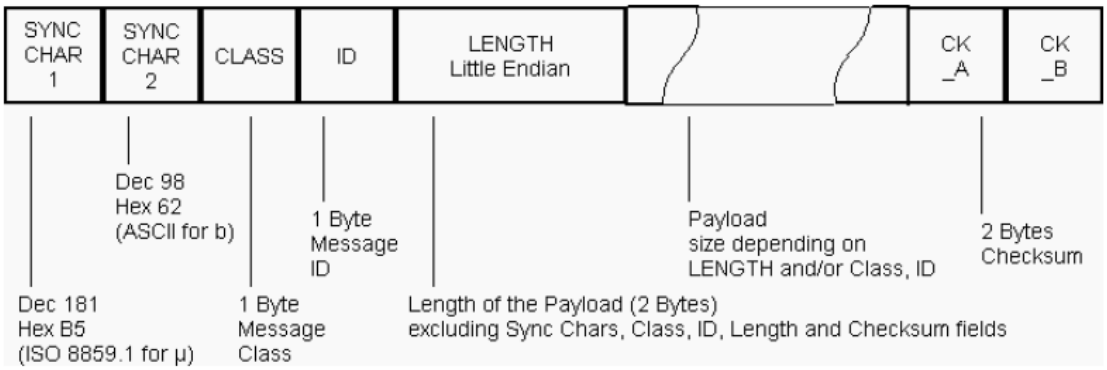
Luvussa 3.2.3 u-blox EVK-7P -vastaanottimen esittelyn yhteydessä huomattiin, että vastaanotin voitaisiin liittää Raspberry Pi:n monella eri tavalla. Tässä työssä vastaanotin päätettiin liittää Raspberry Pi:hin USB-kytkennällä, jolloin se ei tarvitse omaa erillistä virtalähdettä. GNSS-vastaanotin saa virtansa samasta virtalähteestä, kuin Raspberry Pi. USB-kytkennän lisäksi vastaanottimeen on kiinnitettävä vielä erillinen satelliittiantenni, jonka avulla se voi vastaanottaa paikannussatelliittien navigointiviestejä.

EVK-7P aloittaa normaalin toimintansa heti, kun siihen kytketään virta eikä se tarvitse erillistä käynnistämiskäskyä. Käynnistyttyään vastaanotin aloittaa navigointisatelliittien etsimisen ja oman tilatietonsa välittämisen USB-väylän ylitse. Vastaanotin välittää satelliiteilta tulevan paikannustiedon NMEA-standardin mukaisessa muodossa [31][45, p. 48] ja oman tilatietonsa u-bloxin omassa UBX-tietoformaatissa [45, p. 73]. Kuvassa 4.5 on esitelty NMEA- ja UBX-viestikehysten rakenne.

EVK-7P:stä löytyy paljon erilaisia ominaisuuksia ja toiminnallisuuksia, kuten esimerkiksi lokitiedostojen tallentaminen laitteen omalle flash-muistille, sekä mahdollisuus seurata GPS satelliittien lisäksi myös Galileo tai GLONASS satelliitteja. EVK-7P:n tallennuskapasiteetti on kuitenkin varsin pieni ja onkin käytännöllisempää tallentaa vastaanottimelta luettu data suoraan Raspberry Pi:n muistikortille.



(a) NMEA-viestin rakenne [45, p. 48]



(b) UBX-viestin rakenne [45, p. 73]

Kuva 4.5 *Satelliittipaikantimen käyttämät viestiformaatit [45]. NMEA on yleisesti navigointisovelluksissa käytetty standardi [31] ja UBX on u-bloxin oma viestiformaatti [45].*

5. OHJELMISTOKUVAUS

Raspberry Pi:lle toteutettiin Bluetooth-palvelin, jonka avulla järjestelmää voidaan ohjata. Rajapinnan avulla mittaukset voidaan aloittaa, ajastaa ja lopettaa. Tämän lisäksi ohjelmiston kautta voidaan katsoa mittauksen aikana, kuinka kauan mittaus on kestänyt, paljonko sitä on vielä jäljellä ja kuinka paljon dataa mittauksen aikana on kustakin laitteesta kerätty. Ohjelmiston UML-kaavio löytyy liitteestä A.

Palvelinohjelmisto toteutettiin Python-ohjelmointikielellä [36]. Syy tähän valintaan on Bluetooth ja USB ohjelmoinnin alustariippumattomuus PyBluez ja PySerial kirjastoilla [35][11], jotka mahdollistivat ohjelmiston kehittämisen ja testaamisen Windows-ympäristössä.

5.1 Konfiguraatiotiedosto ja Bluetooth-rajapinta

Ohjelmaa käytetään sen Bluetooth-rajapinnan kautta. Rajapinnan kautta voidaan valita mittauksessa käytettävä konfiguraatiotiedosto, aloittaa ja lopettaa mittaaminen, sekä selvittää järjestelmän nykyinen tila. Tässä luvussa esitellään ohjelman käyttämän konfiguraatiotiedoston rakenne, sekä Bluetooth-rajapinta.

5.1.1 Konfiguraatiotiedosto

Portinlukijoiden luonti ja määrittely perustuu JSON-pohjaisiin konfiguraatiotiedostoihin, jotka täytyy manuaalisesti luoda palvelimelle ennen ohjelman käyttöä. Konfiguraatiotiedoston avulla määritetään, kuinka kauan mittausjakso kestää, sekä mittauksessa käytettyjen USB-porttien asetukset. Listauksessa 5.1 on esimerkki konfiguraatiotiedostosta, jossa KVH 1750 ja EVK-7P käynnistetään yhden tunnin ajaksi.

```
{
  "runtime": 3600,
  "ports": [
      {
      "id": "kvh",
      "portname": "/dev/ttyUSB0",
      "baudrate": 921600,
      "filename": "kvh.bin"},
      {
      "id": "ublox",
      "portname": "/dev/ttyACM0",
      "baudrate": 9600,
      "filename": "uBlox.ubx"}
  ]
}
```

Listaus 5.1 *Esimerkki konfiguraatiotiedosto, joka käynnistää tunnin pituisen mittauksen. Mittauksessa on mukana kaksi USB-porttia: /dev/ttyUSB0 (KVH 1750) ja /dev/ttyACM0 (EVK-7P). Mittauksesta kerätty mittausdata tallennetaan tiedostoihin* kvh.bin *ja* uBlox.ubx

JSON-tiedoston formaatti täytyy olla validi ja siitä täytyy löytyä kaikki esimerkissä 5.1 olevat kentät. `runtime`-kenttä määrittää, kuinka kauan portteja luetaan. Mittauksen pituus ilmaistaan sekunneissa, joten esimerkiksi `runtime`:n arvo 3600 tarkoittaa tunnin pituista mittausajanjaksoa. USB-porttien asetukset ovat taulukossa `ports`, jonka kukin tietue määrittää yhden portinlukijan. Jokaisesta tietueesta on löydyttävä portin tunnus (`portname`), siirtonopeus (`baudrate`), sekä tiedostonimi (`filename`), johon portista luettu data kirjoitetaan.

Konfiguraation kentät ovat kestoa ja siirtonopeutta lukuun ottamatta tyypiltään merkkijonoja. Kesto ja siirtonopeus ovat vuorostaan tyypiltään numeroita. Listauksessa 5.1 portin osoite on unixille tyypillisesti `/dev/`-hakemistossa oleva tiedosto, mutta porttiin voidaan viitata myös sen `COM`-osoitteella, jos ohjelmaa ajetaan Windows-koneessa.

5.1.2 Bluetooth-rajapinta

Ohjelman komennot ovat `load`, `start`, `stop status`, `help` ja `quit`. Seuraavissa taulukoissa on dokumentoitu kunkin komennon toiminta.

Konfiguraation lataus	
Komento	`load <filename>`
Parametrit	`<filename>` Konfiguraatiotiedoston nimi
Kuvaus	Lataa annetun konfiguraatiotiedoston ja laittaa ohjelman valmiiksi aloittamaan konfiguraation mukaisen mittauksen. Latauksen jälkeen mittauksen voi aloittaa `start`-komennolla. Ladattua konfiguraatiota voidaan tarkastella myöhemmin `status`-komennolla.

Mittauksen aloitus	
Komento	`start [<filename>]`
Parametrit	`[<filename>]` Valinnainen: Konfiguraatiotiedoston nimi.
Kuvaus	Käynnistää mittaamisen. Komentoa varten pitää joko ladata konfiguraatiotiedosto etukäteen `load`-käskyllä, tai antaa parametrinä konfiguraatiotiedoston nimi. Jos komennon yhteydessä annetaan konfiguraatiotiedosto, käsky aluksi lataa konfiguraatio tiedoston ja aloittaa mittaamisen tämän jälkeen mikäli mahdollista. Mittaus jatkuu konfiguraatiossa annetun ajan, ellei sitä lopeteta `stop`-komennolla ennen sitä.

Mittauksen lopettaminen	
Komento	`stop`
Parametrit	-
Kuvaus	Lopettaa käynnissä olevan mittauksen.

Portinlukijoiden tila	
Komento	`status`
Parametrit	-
Kuvaus	Tulostaa listauksen, jossa on lueteltu kaikkien ladattujen portinlukijoiden tiedot. Listauksesta käy ilmi kunkin USB- tai sarjaportin osoite, niiden tiedonsiirtonopeudet, kuinka kauan se on ollut päällä, kuinka kauan lukijaa aiotaan käyttää, minkä nimiseen lokitiedostoon data tallennetaan ja mikä on lokitiedoston koko.

Ohjeet	
Komento	`help`
Parametrit	-
Kuvaus	Tulostaa rajapinnan ohjeet.

Yhteyden sulkeminen	
Komento	`quit`
Parametrit	-
Kuvaus	Sulkee tämän Bluetooth yhteyden. Yhteyden sulkeminen ei lopeta käynnistettyä mittausta. Yhteys voidaan katkaista myös ilman tätä komentoa.

Kaikki edellä luetellut komennot ovat ASCII-muotoista selkotekstiä ja niiden loppuun vaaditaan rivinvaihtomerkki `LF`. Käyttäjä ei kuitenkaan voi lähettää uutta käskyä, ennen kuin palvelin on käsitellyt edelliseen käskyn. Käyttäjä ei myöskään voi lähettää enempää kuin yhden käskyn kerrallaan.

5.2 Käytetyt kirjastot ja avainkomponentit

Ohjelmiston toteutuksen kannalta tärkeimmät ulkoiset riippuvuudet ovat python-kieleen sisäänrakennettu moniprosessointi moduuli, sekä PyBluez ja PySerial -kirjastot.

5.2.1 Bluetooth-kirjasto PyBluez

PyBluez on Bluez-rajapinnan toteutus pythonille, joka Linuxin lisäksi toimii myös Windows -käyttöjärjestelmissä [35]. Kirjastoa käytetään ohjelmassa RFCOMM-portin luomiseen ja sitä vastaavan palvelinsoketin (*socket*) käyttämiseen. RFCOMM-protokolla

on hyvin samankaltainen, kuin TCP (engl. Transmission Control Protocol), sillä kummatkin takaavat luotettavan kommunikaatioväylän [6]. Näin ollen RFCOMM-soketin ohjelmoinnissa pätevät samat periaatteet kuin TCP-sokettien ohjelmoinnissa.

5.2.2 Rinnakkaisuus moniprosessointi-moduulilla

Moniprosessointi (engl. Multiprocessing, [36, ch. 17.2]) on python-kielen ominaisuus, jonka avulla voidaan luoda itsenäisiä prosesseja. Moniprosessointi moduulin prosessien suurin etu pythonin säikeisiin (engl. Thread, [36, ch. 17.1]) verrattuna on pythonin *globaalin tulkkilukon* (engl. Global Interpreter Lock, GIL) kiertäminen. Normaalisti tulkkilukko varmistaa, että python-tulkki voi tietyllä ajan hetkellä suorittaa vain yhtä säiettä kerrallaan [36]. Moniprosessointi-moduuli ohittaa tämän rajoitteen luomalla erillisiä prosesseja, jotka eivät ole riippuvaisia toisten python-prosssien tulkkilukkojen toiminnasta.

5.2.3 USB-porttien lukeminen PySerial-kirjastolla

PySerial on Chris Liechtin kehittämä python-moduuli, jonka avulla voidaan käyttää käyttöjärjestelmän sarja- ja USB-portteja [11]. PySerial:n tärkein osa on *Serial*-olio, jonka avulla portteja voidaan luoda, lukea, kirjoittaa ja sulkea. Ohjelman USB-porttien hallinta onkin pitkälti PySerial moduulin Serial -olioiden vastuulla.

5.3 Yksityiskohtainen toimintakuvaus

Seuraavaksi käydään lävitse vaihe vaiheelta, kuinka Bluetoothin kautta lähetetty viesti tulkataan oikeaksi lokimanagerin funktiokutsuksi ja kuinka tämän hallinnoimat portinlukijat toimivat.

5.3.1 Bluetooth-soketit

Bluetooth-sokettejen ohjelmointi on hyvin samanlaista kuin esimerkiksi TCP-sokettien ohjelmointi: pääohjelmassa luodaan PyBluez kirjaston tarjoama BluetoothSocket, joka konfiguroidaan RFCOMM-protokollan mukaiseksi ja asetetaan kuuntelemaan

mahdollisia Bluetooth-yhteydenottoja. Kun Bluetooth-yhteys on muodostettu, luodaan tätä varten erillinen asiakassoketti. Ohjelma siis muodostaa uusia yhteyksiä palvelinsoketin avulla ja kommunikoi eri asiakkaiden kanssa asiakassokettejen kautta.

Yhteyden muodostamisen jälkeen ohjelma siirtyy palvelinsilmukkaan, jossa asiakassoketin kautta vastaanotettuja viestejä lähetetään edelleen komentojäsentäjälle. Komentojäsentäjän funktioista tulee kaksi paluuarvoa: ensimmäinen kertoo, onnistuiko komento niin kuin piti ja toinen sisältää merkkijonon, joka kuvastaa palvelimen tilaa. Tämä merkkijono lähetetään asiakkaalle vastaukseksi. Listauksessa 5.2 on esimerkki, miltä kommunikointi palvelimen kanssa näyttää käyttäjän näkökulmasta.

```
# Yhteys muodostettu
> load foobar.json
error: File not found
> load config.json
success: Configs loaded from config.json. 2 loggers prepared
> start
success: 2 loggers started
> quit
Quit command received. Closing connection.
# Yhteys suljettu
```

Listaus 5.2 *Esimerkki Bluetooth kommunikoinnista. Esimerkissä yritetään aluksi ladata* `foobar.json`*-niminen konfiguraatiotiedosto, mutta tämän epäonnistuttua yritetään ladata* `config.json` *mukainen konfiguraatio. Alustuksen onnistuttua konfiguraation mukaiset portinlukijat käynnistetään ja yhteys palvelimelle suljetaan. Portinlukijat jatkavat toimintaansa yhteyden sulkemisesta huolimatta.*

Palvelinsilmukkaa jatketaan niin kauan, kunnes asiakas sulkee yhteyden tai yhteys havaitaan katkenneeksi. Palvelin palvelee vain yhtä asiakasta kerrallaan ja jos toinen Bluetooth-laite yrittää muodostaa yhteyden, palvelin ei vastaa uuteen yhteydenottopyyntöön.

5.3.2 Komentojäsentäjä

Komentojäsentäjän tehtävänä on tulkata ohjelmalle annetut käskyt ja tarkistaa onko komento tai sen parametrit lailliset. Jos annettu komento on *help*, komento on

tuntematon tai komennolla väärä määrä parametreja, ei jäsentäjän tarvitse kutsua lokimanageria ja se vastaa itse annettuun käskyyn. Muussa tapauksessa jäsentäjä kutsuu käskyä vastaavaa lokimanagerin funktiota.

Lokimanagerin funktioiden paluuarvo on aina monikko, jonka ensimmäisestä arvosta nähdään, onnistuiko käsky vai ei. Monikon toinen arvo on ihmisluettava merkkijono, joka kertoo mitä käskyä suorittaessa tapahtui. Nämä tiedot vuorostaan välitetään eteenpäin soketille, jonka kautta ne välitetään edelleen asiakaslaitteelle.

5.3.3 Lokimanageri

Lokimanagerin tehtävänä on valvoa ja ohjata USB-portteja lukevien prosessien toimintaa. Lokimanageri on vastuussa portinlukijoiden luonnista, käynnistämisestä ja sulkemisesta. Lisäksi portinlukijoiden ajonaikaiset tiedot, kuten esimerkiksi aktiivisten lukijoiden määrä ja näiden käyttämien tiedostojet koot, voidaan hakea lokimanagerin kautta. Portinlukijoiden elinkaaret ovat vahvasti sidottuja tähän luokkaan.

Portinlukijat alustetaan *load_ config*-funktion avulla, joka lukee annetun konfiguraatiotiedoston. Alustamisen jälkeen portinlukijat voidaan käynnistää *start_ logging*-funktiolla, joka yrittää käynnistää portinlukijat. Jos yhden tai useamman lukijan käynnistämisessä ilmenee ongelmia, välitetään käyttäjälle virheilmoitus. *start_ logging*-funktiolle voidaan vaihtoehtoisesti myös antaa käytettävä konfiguraatiotiedosto, jolloin manageri korvaa vanhan konfiguraation uudella. Kun portinlukijat on käynnistetty, ne jatkavat portin lukemista, kunnes konfiguraatiossa määritelty aika on kulunut tai managerin *stop_ logging*-funktiota kutsutaan.

Lukijoiden tilatiedot voidaan selvittää *list_ loggers*-funktion avulla riippumatta siitä, ovatko lukijat käynnissä vai ei. Funktio hakee kunkin portinlukijan tiedot näiden *get_ logger_ info*-funktion avulla ja yhdistää vastaukset yhdeksi listaksi. Listauksesta saadaan selville portin konfiguraatio, kuinka pitkäksi ajaksi se on ajastetty, kuinka kauan se on ollut käynnissä, sekä kuinka iso sen lokitiedosto on. Listauksesta 5.3 löytyy esimerkki, miltä lokimanagerin antama listaus voisi näyttää, kun listauksessa 5.2 käynnistettyä mittausta on kulunut noin 11 minuuttia.

```
# Yhteys muodostettu
> status
success: List of loggers:
/dev/ttyUSB0,921600,kvh.bin,670/3600,23.0MB
/dev/ttyACM0,9600,uBlox.ubx,670/3600,5.8MB
> stop
success: Stop command received. 2 loggers stopped
> quit
Quit command received. Closing connection.
# Yhteys suljettu
```

Listaus 5.3 *Portinlukijoiden tilatiedot. Noin 11 minuutin kuluttua käyttäjä kysyy järjestelmän tilaa* `status`*-komennolla ja lopettaa mittaamisen* `stop`*-komennolla. Tämän jälkeen yhteys suljetaan* `quit`*-komennolla.*

5.3.4 Portinlukijat

Varsinainen USB-porttien lukeminen on portinlukijoiden vastuulla. Lukijat ovat Pythonin standardikirjaston moniprosessointimoduulin Prosessi-olioista (engl. *Process*) periytettyjä, joiden `run`-metodiin on toteutettu USB-porttia lukeva rutiini. *run*-funktiossa Portinlukija kirjoittaa kaiken lukemansa datan suoraan lokitiedostoon, joka sille on konfiguraatiotiedostossa määritelty. Funktio ei ota kantaa siihen, minkälaista dataa USB-portin kautta luetaan.

Portinlukijoihin toteutettiin myös muutama funktio, joita alkuperäisessä Process oliossa ei ole. Portinlukijoilla on esimerkiksi *Multiprocessing.Event* tyyppinen sisäinen lippu *_flag*, jonka avulla *run*-funktion silmukka voidaan pysäyttää hallitusti. Lipun voi asettaa portinlukijan *stop* funktiolla. Tämän lisäksi portinlukijan kautta voidaan selvittää lokitiedoston koko komennolla *get_filesize* ja lukijaan liittyvät tiedot komennolla *get_logger_info*.

Portinlukijat ovat toiminnaltaan suoraviivaisia ja ne toimivat hyvin, kun luettava portti aloittaa datan lähettämisen heti käynnistymisensä jälkeen. Toteutuksen selvä heikkous on, ettei sen kautta voida lähettää viestejä laitteelle. Nykyisellä toteutuksella ei siis käytännössä voida lukea laitteita, jotka vaativat erillisen käynnistyskomennon. KVH 1750 ja EVK-7P eivät ole tällaisia laitteita, mutta asia on hyvä ottaa huomioon järjestelmän jatkokehityksessä.

6. ESIMERKKEJÄ

Tässä luvussa esitellään työssä suunnitellun mittauslaitteiston sovelluskohteita. Luvussa osoitetaan laitteiston inertiamittausyksikkö riittävän tarkaksi, jotta sillä voidaan havaita maapallon pyörimisliike. Lisäksi luvussa kokeillaan, miten laitteisto soveltuu pienmetsäkoneessa suoritettaviin inertiamittauksiin.

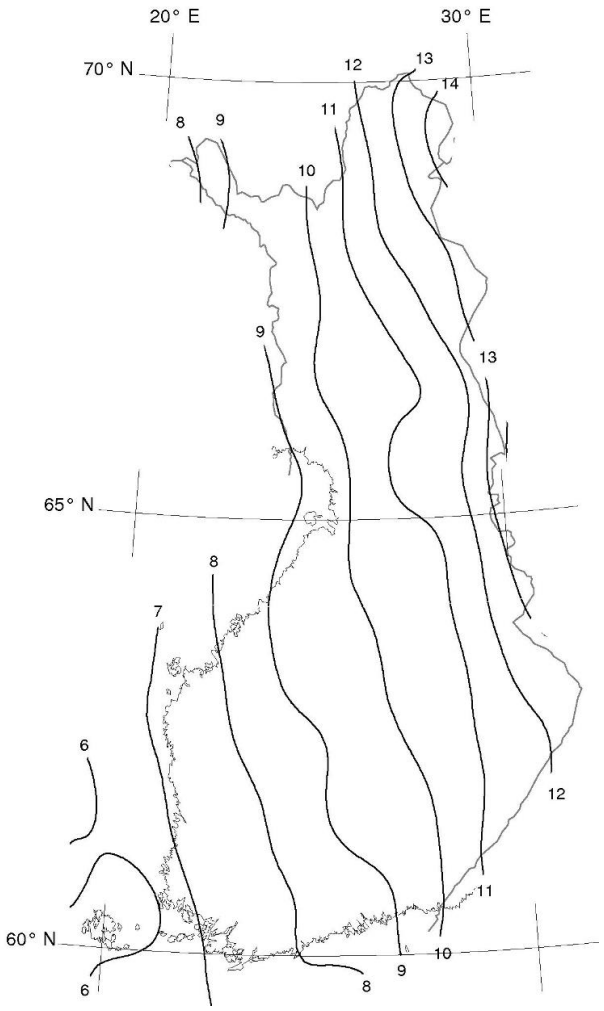
6.1 Napapohjoisen löytäminen

Napapohjoisella tarkoitetaan suuntaa maapallon pohjoisnavalle, joka on eri kuin kompassin näyttämä magneettinen pohjoinen. Kompassineulan ja napapohjoisen välistä erotusta kutsutaan erannoksi. Ilmatieteen laitoksen mukaan *"Suomessa eranto vaihtelee Ahvenanmaan 6 asteesta itärajan 14 asteeseen"* [24]. Kuvassa 6.1 on Suomen erantokartta, josta nähdään erannon suuruus eri puolella suomea.

Luvussa 3.2.2 esitettiin vaatimus, että laitteiston inertiamittausyksikkö olisi riittävän tarkka todellisen pohjoisen löytämiseen. Tätä vaatimusta testattiin toteuttamalla artikkelissa [8] esitelty napapohjaisen haku-algoritmin reaaliaikainen visualisaatio. Sovelluksella haluttiin erityisesti nähdä, kuinka hyvin todellinen pohjoinen voidaan löytää KVH 1750:sta kerätyllä suodattamattomalla raakadatalla. Toteutuksessa oletetaan inertiamittausyksikön olevan pöydällä z-akseli painovoiman suuntaisesti.

6.1.1 Ohjelman toteutus

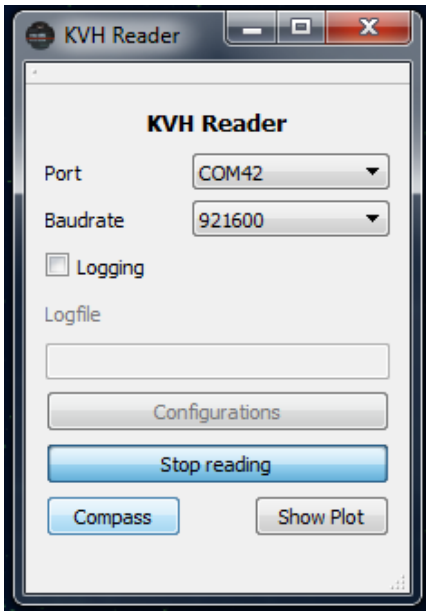
Koska työssä suunniteltu järjestelmä oli suunniteltu mittausdatan keräämistä varten, eikä niinkään datan visualisointia varten, ei pohjoisen löytämistä toteutettu työssä esitellylle alustalle. Sen sijaan algoritmin visualisointi tehtiin Windows työpöytäsovelluksena. Ohjelman toteutettiin C++-kielellä käyttäen Qt-ohjelmistokehystä [37], sekä QCustomPlot-datavisualisointi kirjastoa [16].



Kuva 6.1 *Erannon suuruus eripuolilla Suomea. Kuva: [24].*

Ohjelma käynnistyy päänäkymään (kuva 6.2), josta käsin käyttäjä valitsee aluksi USB-portin asetukset, eli KVH 1750:llaa vastaavan USB-portin ja sen tiedonsiirtonopeuden. Tämän jälkeen käyttäjä avaa kompassinäkymän, joka visualisoi algoritmin löytämää napapohjoista mittausyksikön suhteen. Näkymän yhteydessä käynnistyy 50 ms viiveellä toimiva ajastin, joka päivittää pohjoisen suunnan 20 kertaa sekunnissa.

Pohjoisen etsiminen alkaa laskemalla gyroskoopin ja kiihtyvyysanturin lukemien keskiarvo halutun pituiselta ajalta, jonka kokoa voidaan säätää näkymän alaosan säätimellä. Keskiarvon laskemisen jälkeen saadut lukemat normalisoidaan yksikkövektoreiksi $\bar{m}_w$ ja $\bar{m}_a$, missä $\bar{m}_w$ on gyroskoopin normalisoitu yksikkövektori ja $\bar{m}_a$ on



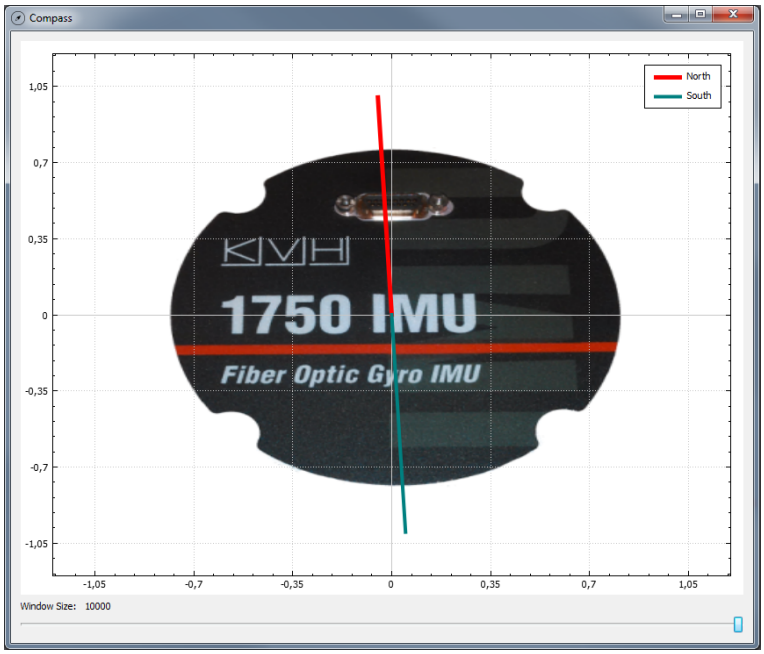
Kuva 6.2 *Ohjelman päänäkymä. KVH 1750:lle löytyy portista* `COM42` *ja sen baudinopeus on 921.6 kBd. Kompassinäkymä avautuu valitsemalla* `Compass`*.*

kiihtyvyysanturin normalisoitu yksikkövektori.

Napapohjoisen määrittämistä varten inertiamittausyksikköön sidottu koordinaatisto (engl. body frame) täytyy muuttaa paikalliseen koordinaatistoon (engl. local frame), jonka x-akseli osoittaa pohjoiseen, y-akseli länteen ja z-akseli alas. Tämä tehdään artikkelissa [8] esitetyn suuntakosinimatriisin

$$C_L^B = \begin{pmatrix} \frac{\bar{m}_{wx} - (\bar{m}_{ay}\sin(\lambda))}{\cos(\lambda)} & \frac{\bar{m}_{wy} - (\bar{m}_{ay}\sin(\lambda))}{\cos(\lambda)} & \frac{\bar{m}_{wz} - (\bar{m}_{az}\sin(\lambda))}{\cos(\lambda)} \\ \frac{(\bar{m}_{ay}\bar{m}_{wz} + \bar{m}_{az}\bar{m}_{wy})}{\cos(\lambda)} & \frac{(\bar{m}_{az}\bar{m}_{wx} - \bar{m}_{ax}\bar{m}_{wz})}{\cos(\lambda)} & \frac{(\bar{m}_{ax}\bar{m}_{wy} - \bar{m}_{ay}\bar{m}_{wx})}{\cos(\lambda)} \\ \bar{m}_{ax} & \bar{m}_{ay} & -\bar{m}_{az} \end{pmatrix} \tag{6.1}$$

avulla. Matriisin λ tarkoittaa mittauspaikan leveyspiiriä. Esimerkiksi Tampereen teknillisen yliopiston kampuksen sijainti on noin 61.450° pohjoista leveyttä WGS 84 (*World Geodetic System*) -koordinaatistossa. Napapohjoisen suunta mittausyksikön suhteen vuorostaan saadaan kertomalla suuntakosinimatriisi pohjoisen suunnalla paikallisessa koordinaatistossa. Koska pohjoisen suunnaksi oli määritelty x-akseli,



Kuva 6.3 *Kompassinäkymä. Punainen viiva osoittaa pohjoiseen ja vihreä viiva osoittaa etelään.*

pitää C_L^B kertoa vektorilla $[\mathbf{1\ 0\ 0}]^\mathbf{T}$. Napapohjoisen sijainti saadaan siis yhtälöstä

$$C_L^B \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} \dfrac{\bar{m}_{wx} - (\bar{m}_{ay}\sin(\lambda))}{\cos(\lambda)} & \dfrac{(\bar{m}_{ay}\bar{m}_{wz} - \bar{m}_{az}\bar{m}_{wy})}{\cos(\lambda)} & \bar{m}_{ax} \end{bmatrix}^T, \tag{6.2}$$

missä vektorin ensimmäinen ja toinen termi ovat napapohjoisen suunta mittausyksiköstä käsin katsottuna. Viimeisen termin $\bar{m}_{ax}$ avulla voidaan laskea anturin kallistuskulma, sillä anturin ollessa levossa kiihtyvyysanturin x-komponentissa näkyisi vain painovoimasta johtuva kiihtyvyys. Kaksiulotteisessa visualisaatiossa tätä termiä ei kuitenkaan tarvita.

Koska visualisoinnin x- ja y-akseli eivät ole samansuuntaiset koordinaatiston x- ja y-akseleiden kanssa, täytyy kaavasta 6.2 saadut x- ja y-koordinaatit muuttaa vielä graafisen esityksen koordinaatistoon. Tämä tehdään kääntämällä x- ja y-akseleita $3\pi/4$ radiaania myötäpäivään. Kuvassa 6.3 on lopullinen visualisaatio, jossa pohjoinen ja etelä on piirretty suhteessa inertiamittausyksikön asentoon.

6.1.2 Toteutuksen arviointi

Sovellus pystyy löytämään napapohjoisen suunnan alle viidessä sekunnissa, mutta näin lyhyellä aikaikkunalla suunta on melko epävakaa ja suunta heittelee todella paljon ympäristön vaikutuksesta. Esimerkiksi pöytään nojaaminen tai jopa pöydän ohitse menevästä ohikulkijasta johtuva tärinä riittävät häiritsemään ratkaisun tarkkuutta. Ratkaisusta saadaan vakaampi kasvattamalla aikaikkunan kokoa, mutta tämän kääntöpuolena on uudelleen löytämisen hidastuminen. Jos mittausyksikön asentoa muutetaan, sillä kestää vähintään ikkunan koon pituinen aika napapohjoisen uudelleen löytämiseen. Tärinästä johtuva virhe on kuitenkin mahdollista poistaa edistyneemmän suodatusalgoritmin avulla [39].

Sovellus täyttää ensisijaisen suunnittelutavoitteensa. Sovellus osoittaa, että KVH 1750:lla voidaan havaita maapallon pyöriminen, josta vuorostaan voidaan laskea napapohjoisen suunta. Tätä tietoa voidaan käyttää esimerkiksi ajoneuvon asennon selvittämiseen tietyllä ajanhetkellä.

6.2 Pienmetsäkoneen tärinän mittaaminen

Kun pienmetsäkoneella hoidetaan taimikkoa, sen osat joutuvat kestämään monenlaista rasitusta. Varsinkin koneen puomisto on kovilla, kun se joutuu työlaitteen painon lisäksi kestämään raivauksesta johtuvaa tärinää ja tärähdyksiä. Joskus tärinän ja painon yhteisvaikutus voi olla riittävän voimakas murtamaan leikkuria kannattelevan metallipuomin. Jotta puomi voitaisiin suunnitella kestämään työskentelystä johtuvaa rasitusta, on tiedettävä minkälaisia voimia puomiin kohdistuu työskentelyn aikana. Puomiin kohdistuvan rasituksen määrä riippuu monesta tekijästä. Rasituksen määrään vaikuttavat leikattavan puun ominaisuudet, leikkurin ominaisuudet, sekä koneen käyttöön liittyvät yksityiskohdat.

Tämän työn yhteydessä suoritettiin mittaus, jossa kerättiin tietoa suomalaisen pienmetsäkonevalmistajan Usewood Frest Tec Oy:n *Forest Master* -pienmetsäkoneen käytöstä (kuva 6.4) [48]. Mittauksessa pienmetsäkoneen puomistoon kiinnitettiin joukko MEMS-antureita, joilla mitattiin puomistoon kohdistuvia voimia. Diplomityössä suunniteltua laitteistoa käytettiin mittauksissa referenssilaitteena, johon muista antureista saatua mittausdataa verrattiin. Testiajot suoritettiin Usewoodin koemetsässä Muuramessa.

Kuva 6.4 *Usewood Forest Master -pienmetsäkone [48]*

6.2.1 Mittausasetelma

Metsäkoneen puomien tärinää mittaavat anturit oli asennettu eri puolille metsäkoneen puomistoa. Mittauksessa käytettiin iNemo pohjaisia inertiamittausyksiköitä [34] sekä iSense Sk Wheelsensor inertiamittausyksiköitä [26]. iNemo ja iSense SK Wheelsensor antureiden tehtävänä oli mitata tärinän määrää puomin eri osissa eri työvaiheiden aikana ja KVH 1750:n tarkoitus oli toimia referenssianturina metsäkoneen ohjaamossa. Mittauksissa vertailtiin koneen eri osien tärinää sekä koneen tärinäprofiilia kahden eri kuljettajan välillä.

Anturit oli aseteltu koneeseen seuraavasti: KVH 1750 oli kiinnitetty ohjaamoon ja ohjaamon katolle oli kiinnitetty GPS-702-GG antenni [33]. Antenniin oli kytketty luvussa 3.2.3 esitelty NovAtelin DL-4 Plus GPS-vastaanotin. iNemo anturit oli aseteltu ohjaamon ja etuosan väliseen niveleen, sekä metsäkoneen 2. ja 3. puomiin. iSense SK Wheelsensor-anturit oli aseteltu ensimmäiseen puomiin, sekä raivainta kannattelevaan puomiin. Kuvassa 6.5 on esitetty antureiden sijainti metsäkoneessa.

Kuva 6.5 *Anturit metsäkoneessa. Metsäkoneen ohjaamossa (**1**) oli KVH 1750 ja NovAtel DL-4 Plus. DL-4 Plus käytti metsäkoneen katolle kiinnitettyä GPS antennia (**2**). iNemot (**3**,**4** ja **5**) ja iSense SK Wheelsensor (**6** ja **7**) anturit oli aseteltu eri puolille koneen puomistoa.*

6.2.2 Laitteiston soveltuminen käyttökohteeseen

Järjestelmän asentaminen metsäkoneeseen oli helppoa ja mittalaitteilla saatiin kerättyä eheä datasetti myöhempää analyysiä varten. Järjestelmä osoittautui hyväksi ja luotettavaksi mittausjärjestelmäksi. Mittauksissa kerätyn anturidatan Fourier-muunnokset sekä spektrogrammit löytyvät liitteestä C.

Wheelsensor ja KVH 1750:n datoja vertailemalla huomattiin, että metsäkoneessa ei esiintynyt merkittäviä voimia 500 Hz ja 1000 Hz välillä. Toisin sanoen KVH 1750:n mittausalue oli sopiva mittaamaan metsäkoneessa esiintyvää tärinää, eikä siltä jäänyt tärkeitä tärinä-taajuuksia huomaamatta. Tämän perusteella voidaan myös sanoa, ettei iNemo antureiden mittaustaajuus ollut riittävä pienmetsäkoneen tärinän mittaamiseen. Jos pienmetsäkoneessa esiintyvä tärinä on suurimmillaan 400 Hz taajuista, pitäisi Nyquistin näytteenottoteoreeman mukaan tärinää mittaavien

antureiden taajuuden oltava 800 Hz [41].

Koska KVH 1750 oli asennettu lähelle metsäkoneen moottoria, pystyttiin sen taajuustarkastelun avulla määrittämään metsäkoneen moottorin käyntitaajuudeksi noin 50 Hz, joka vastaa noin 3000 kierrosta minuutissa. Tämän tiedon avulla moottorista johtuva tärinä voidaan erottaa muiden antureiden datasta, joka mahdollistaa leikkurista johtuvan tärinän tarkemman analysoinnin.

Datan käsittelyn yhteydessä huomattiin tarve mittausdatan paremmalle synkronoinnille. Koska jokainen mittauksessa käytetty inertiamittausyksikkö oli käynnistetty eri aikaan ja ne toimivat eri taajuuksilla, oli mittausyksiköillä kerätyn datan synkronointi jälkikäteen hyvin haastavaa. Jatkokehitysideaksi keksittiinkin antureiden keskitetty ja koordinoitu käynnistäminen Raspberry Pi:n kautta. Esimerkiksi mittauksessa käytetyt anturit olisi voitu yhdistää Bluetoothilla Raspberry Pi:hin, jolloin kaikki anturit olisivat myös aloittaneet mittaamisen yhtä aikaa.

7. YHTEENVETO

Tässä työssä suunniteltiin inertiamittauslaitteisto, johon on liitetty myös GNSS-vastaanotin sijainnin määrittämistä varten. Laitteisto koostuu Raspberry Pi 2 -pienoistietokoneesta, KVH 1750 -inertiamittausyksiköstä, u-blox:n EVK-7P satelliittivastaanottimesta, Bluetooth-vastaanottimesta, sekä näiden akuista. Laitteiston tarkoitus on toimia referenssi- ja kalibrointilaitteistona halvemmille ja epätarkemmille inertiamittausyksiköille. Laitteistoa tullaan käyttämään kenttäoloissa, jonka vuoksi siitä tehtiin helposti siirreltävä.

Laitteistolle tehtiin python-ohjelmointikielellä ohjelmisto, jonka avulla Raspberry Pi:n kiinnitetyistä laitteista voidaan lukea dataa lokitiedostoihin. Laitteistoa voidaan ohjata sen Bluetooth-rajapinnan kautta etäältä. Tämä on erittäin hyödyllistä, kun mittauslaitteisto on asennettu paikkaan, jossa siihen ei pääse helposti käsiksi mittauksen aikana.

Laitteistoon valittu inertiamittausyksikkö osoitettiin riittävän tarkaksi havaitsemaan maapallon pyörimisliike. Työn yhteydessä toteutettiin pohjoisenhaku-sovellus, joka määrittää napapohjoisen sijainnin maapallon pyörimisen perusteella. Sovellus toteutettiin erillisenä työpöytäsovelluksena käyttäen C++-ohjelmointikieltä ja Qt-ohjelmistokehystä.

Mittausjärjestelmälle havaittiin myös erilaisia jatkokehitysideoita. Järjestelmä olisi esimerkiksi mahdollista saada toimimaan vain yhdellä akulla lisäämällä sopiva muuntaja Cameron Sino-akun ja Raspberry Pi:n väliin. Tämän lisäksi laitteistoa ei ole vielä koteloitu ja sopivan kotelon teettäminen voisikin tehdä järjestelmästä helpommin siirreltävän. Pienmetsäkoneella tehdyissä mittauksissa vuorostaan huomattiin tarve mittauksessa käytettyjen antureiden yhtäaikaiselle käynnistämiselle. Tällöin jokaista anturia ei tarvitsisi kenttäoloissa käynnistää erikseen ja mittausdatan synkronoinnista jälkiprosessoinnissa tulisi helpompaa.

Kaiken kaikkiaan, järjestelmä havaittiin hyödylliseksi mittauslaitteistoksi ajoneu-

voissa tehtäviin inertiamittauksiin. Järjestelmän avulla pystyttiin tarkentamaa MEMS-antureilla tehtyjä havaintoja ja korjaamaan MEMS-antureiden epätarkkuudesta johtuvia mittausvirheitä. Järjestelmä siis täyttää tärkeimmät suunnittelutavoitteensa.

LÄHTEET

[1] "IEEE Standard for Inertial Sensor Terminology," *IEEE Std 528-2001*, 2001.

[2] A. Griffin et al., "Arch linux," 2002-, Saatavissa (viitattu 28.6.2016): https://www.archlinux.org/.

[3] Arch Linux ARM, "Arch linux arm," 2009-, Saatavissa (viitattu 28.6.2016): https://archlinuxarm.org/.

[4] *Matrix-504 Datasheet*, Artila, Saatavissa (viitattu 8.6.2016): http://www.artila.com/en/p_matrix.html.

[5] E. T. Benser, "Trends in inertial sensors and applications," in *Inertial Sensors and Systems (ISISS), 2015 IEEE International Symposium on*, March 2015, pp. 1–4.

[6] *RFCOMM with TS 07.10*, V12 ed., Bluetooth SIG, Nov. 2012, Saatavissa (viitattu 1.8.2016): https://www.bluetooth.com.

[7] "Bluez - official linux bluetooth protocol stack," Bluez Project, Saatavissa (viitattu 29.7.2016): http://www.bluez.org/.

[8] J. Bojja, J. Collin, M. Kirkko-Jaakkola, M. Payne, R. Griffiths, and J. Takala, "Compact north finding system," *IEEE Sensors Journal*, vol. 16, no. 8, pp. 2554–2563, April 2016.

[9] A. Buke, F. Gaoli, W. Yongcai, S. Lei, and Y. Zhiqi, "Healthcare algorithms by wearable inertial sensors: a survey," *China Communications*, vol. 12, no. 4, pp. 1–12, April 2015.

[10] M. Cable, *Calibration: A Technician's Guide*, ser. ISA technician series. ISA, 2005.

[11] "Pyserial," Chris Liechti, Saatavissa (viitattu 2.8.2016): https://pythonhosted.org/pyserial/.

[12] *HTD22110A Instruction Manual*, Clas Ohlson, Apr. 2015.

[13] Debian Project, "Debian - the universal operating system," 1993-, Saatavissa (viitattu 9.6.2016): https://www.debian.org/.

[14] *MIL-DTL-83513F*, Defense Logistics Agency, March 2007.

[15] M. Dumbill, E. Krasnyansky, *sdptool - Linux Man page*, Saatavissa (viitattu 19.8.2016): http://linux.die.net/man/1/sdptool.

[16] E. Eichhammer, "Qcustomplot," Saatavissa (viitattu 12.8.2016): http://qcustomplot.com/.

[17] Fedora Project, "Fedora," 2003-, Saatavissa (viitattu 9.6.2016): https://getfedora.org/.

[18] W. Fink and K. Zak, *agetty Man Page*, Feb. 2016, Saatavissa (viitattu 1.8.2016): http://man7.org/linux/man-pages/man8/agetty.8.html.

[19] *GCC, the GNU Compiler Collection*, Free Software Foundation, 1999, Saatavissa (viitattu 23.6.2016): https://gcc.gnu.org/.

[20] *USB To RS422 Serial Converter Cable Datasheet*, 1st ed., FTDI, Jan. 2010, Saatavissa (viitattu 20.7.2016): http://www.ftdichip.com/Products/Cablee/USBRS422.htm.

[21] S. Gibbs, "Raspberry pi becomes best selling british computer," *The Guardian*, Feb. 2015, Saatavissa (viitattu 28.6.2016): https://www.theguardian.com/technology/2015/feb/18/raspberry-pi-becomes-best-selling-british-computer.

[22] M. Grewal and A. Andrews, "How good is your gyro [ask the experts]," *IEEE Control Systems*, vol. 30, no. 1, pp. 12–86, Feb 2010.

[23] P. D. Groves, *Principles of GNSS, Inertial, and Multisensor Integrated Navigation Systems, Second Edition (Artech House Remote Sensing Library)*. Artech House, 2013.

[24] "Eranto suomessa," Ilmatieteen laitos, 2016, Saatavissa (viitattu 13.9.2016): http://ilmatieteenlaitos.fi/eranto-suomessa.

[25] *iFOG-IMU-1-A Datasheet*, iMar Navigation GmbH, March 2016, Saatavissa (viitattu 28.6.2016): http://imar.de/downloads/FOG-IMU-1-A.pdf.

[26] "iSense Sk Wheelsensor," iSense Sk, Saatavissa rajoitetusti: http://isense-sk.com/wheelsensor/.

[27] E. Kaplan and C. Hegarty, *Understanding GPS: Principles and Applications*. Artech House, 2005.

[28] V. Kempe, *Inertial MEMS: Principles and Practice*, 1st ed. New York, NY, USA: Cambridge University Press, 2011.

[29] *1750 IMU Technical Manual*, KVH Part 54-0858 Rev. F ed., KVH Industries Inc., 2015, Saatavissa rajoitetusti:http://www.kvh.com.

[30] Microsoft, "Windows IoT Core," 2015-, Saatavissa (viitattu 28.6.2016): https://developer.microsoft.com/en-us/windows/iot.

[31] *The NMEA 0183 Protocol*, 2nd ed., National Marine Electronics Association, 1998.

[32] *DL-4plus Datasheet*, NovAtel, 2006, Saatavissa (viitattu 11.8.2016): http://www.novatel.com/support/search/items/Manual.

[33] *GPS-702-GG Datasheet*, NovAtel, 2015, Saatavissa (viitattu 11.8.2016): http://www.novatel.com/support/search/items/Manual.

[34] T. Pihlstrom, "Langaton inertiamittausyksikkö," Master's thesis, Tampereen Teknillinen Yliopisto, 2013.

[35] Piotr Karulis, et al., "Pybluez," Saatavissa (viitattu 2.8.2016): http://karulis.github.io/pybluez/.

[36] "Python," Python Software Foundation, Saatavissa (viitattu 2.8.2016): https://www.python.org/.

[37] "Qt cross-platform software development," Qt Company, Saatavissa (viitattu 12.8.2016): https://www.qt.io/.

[38] *Raspberry Pi 2, Model B Datasheet*, Raspberry Pi Foundation, Adafruit, Saatavissa (viitattu 8.6.2016): https://www.adafruit.com/pdfs/raspberrypi2modelb.pdf.

[39] P. G. Savage, *Strapdown Inertial Navigation Lecture Notes*. Strapdown Associates, INC, Jul. 1997.

[40] Seneca Centre for Development of Open Technology, "Pidora - raspberry pi fedora remix," 2014-, Saatavissa (viitattu 28.6.2016): http://pidora.ca/.

[41] C. E. Shannon, "A mathematical theory of communication," *SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 5, no. 1, pp. 3–55, Jan. 2001.

[42] *SI-opas*, Suomen Standardisoimisliitto, Helmi. 2001.

[43] T. Thompson and P. Green, "Raspbian operating system," 2012-, Saatavissa (viitattu 9.6.2016): https://www.raspbian.org/.

[44] J. Titterton, D. Weston, *Strapdown Inertial Navigation Technology*, ser. Electromagnetics and Radar Series. Institution of Engineering and Technology, 2004.

[45] *u-blox 7 Receiver Description Including Protocol Specification*, V14 ed., u-blox, 2013, Saatavissa (viitattu 29.7.2016): https://www.u-blox.com/en/product/evk-7.

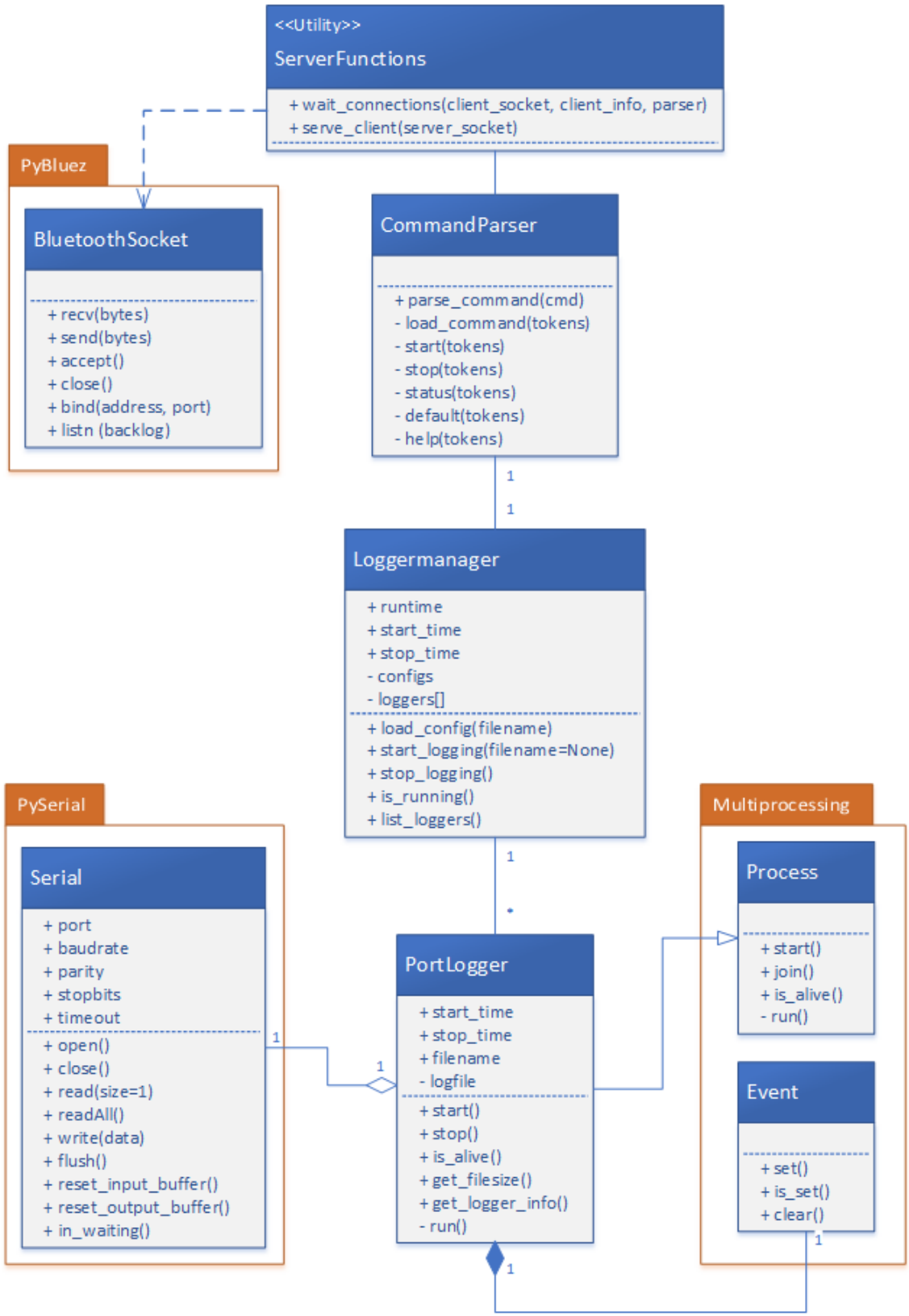
[46] Ubuntu ARM Team, "Ubuntu ARM Project," Saatavissa (viitattu 28.6.2016): https://wiki.ubuntu.com/ARM.

[47] Ubuntu Foundation, "Ubuntu," 2004-, Saatavissa (viitattu 28.6.2016): http://www.ubuntu.com/.

[48] "Usewood forest tec oy," Usewood Forest Tec Oy, Saatavissa (viitattu 4.8.2016): http://www.usewood.fi/index.php/.

[49] P. L. Walter, "The history of the accelerometer," *Sound and Vibration*, pp. 84–92, Jan. 2007.

LIITE A. OHJELMISTON UML-KUVAAJA


<<Utility>>
ServerFunctions
+ wait_connections(client_socket, client_info, parser)
+ serve_client(server_socket)
PyBluez
BluetoothSocket
+ recv(bytes)
+ send(bytes)
+ accept()
+ close()
+ bind(address, port)
+ listn (backlog)
CommandParser
+ parse_command(cmd)
- load_command(tokens)
- start(tokens)
- stop(tokens)
- status(tokens)
- default(tokens)
- help(tokens)
1
1
Loggermanager
+ runtime
+ start_time
+ stop_time
- configs
- loggers[]
+ load_config(filename)
+ start_logging(filename=None)
+ stop_logging()
+ is_running()
+ list_loggers()
1
*
PySerial
Serial
+ port
+ baudrate
+ parity
+ stopbits
+ timeout
+ open()
+ close()
+ read(size=1)
+ readAll()
+ write(data)
+ flush()
+ reset_input_buffer()
+ reset_output_buffer()
+ in_waiting()
1
1
PortLogger
+ start_time
+ stop_time
+ filename
- logfile
+ start()
+ stop()
+ is_alive()
+ get_filesize()
+ get_logger_info()
- run()
1
Multiprocessing
Process
+ start()
+ join()
+ is_alive()
- run()
Event
+ set()
+ is_set()
+ clear()
1

LIITE B. BLUETOOTH RAJAPINTA

Kaikki alla olevat komennot ovat ASCII-muotoista tekstiä. Jokaisen komennon jälkeen on tultava rivinvaihtomerkki `LF` ja komennot on lähetettävä yksi kerrallaan. Seuraavaa komentoa ei voi lähettää ennen, kuin edellinen komento on käsitelty.

Konfiguraation lataus	
Komento	`load <filename>`
Parametrit	`<filename>` Konfiguraatiotiedoston nimi
Kuvaus	Lataa annetun konfiguraatiotiedoston ja laittaa ohjelman valmiiksi aloittamaan konfiguraation mukaisen mittauksen. Latauksen jälkeen mittauksen voi aloittaa `start` komennolla. Ladattua konfiguraatiota voidaan tarkastella myöhemmin `status` komennolla.

Mittauksen aloitus	
Komento	`start [<filename>]`
Parametrit	`[<filename>]` Valinnainen: Konfiguraatiotiedoston nimi.
Kuvaus	Käynnistää mittaamisen. Komentoa varten pitää joko ladata konfiguraatiotiedosto etukäteen `load` käskyllä, tai antaa parametrinä konfiguraatiotiedoston nimi. Jos komennon yhteydessä annetaan konfiguraatiotiedosto, käsky aluksi lataa konfiguraatio tiedoston ja aloittaa mittaamisen tämän jälkeen mikäli mahdollista. Mittaus jatkuu konfiguraatiossa annetun ajan, ellei sitä lopeteta `stop` komennolla ennen sitä.

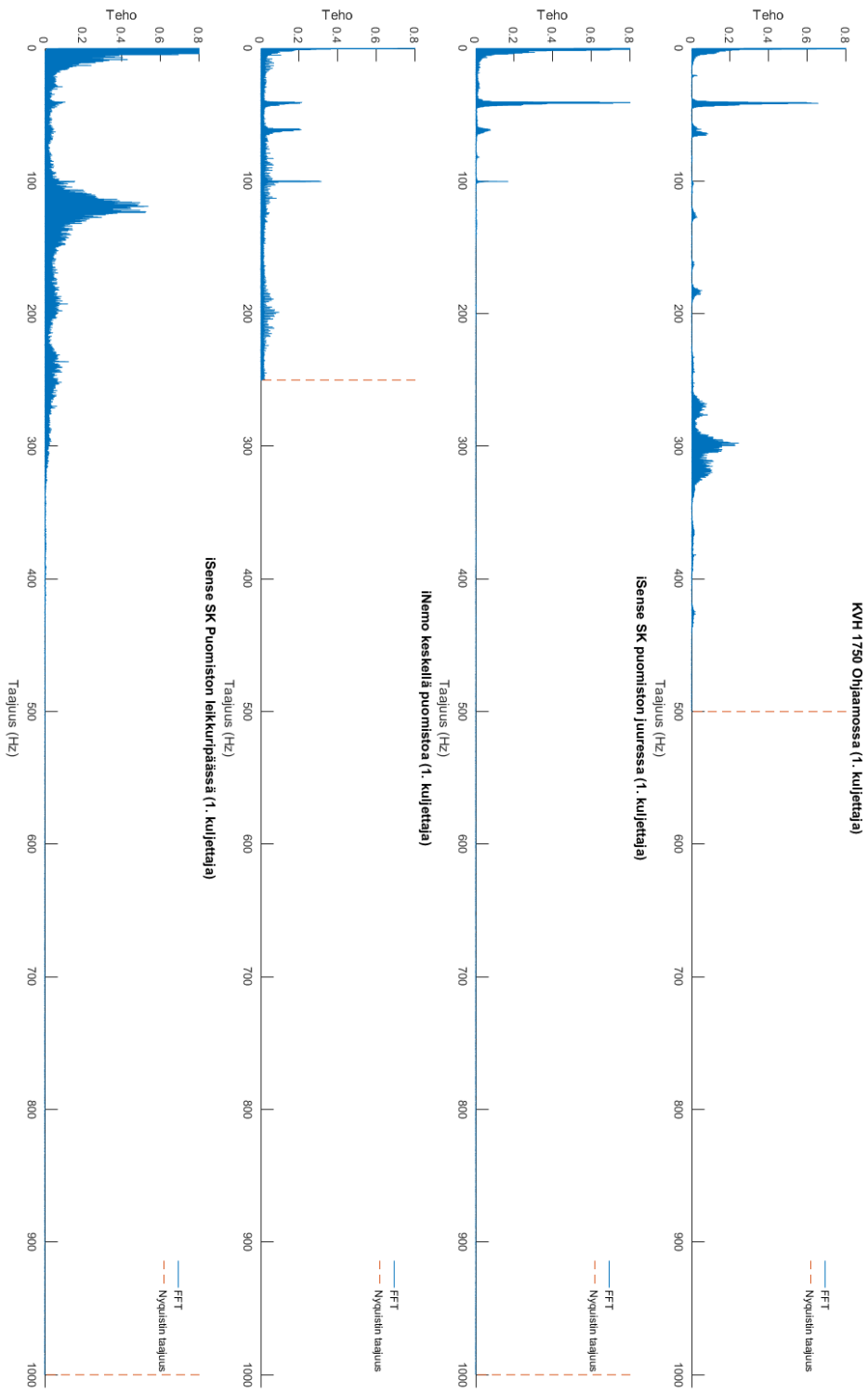
Mittauksen lopettaminen	
Komento	`stop`
Parametrit	-
Kuvaus	Lopettaa käynnissä olevan mittauksen.

Portinlukijoiden tila	
Komento	`status`
Parametrit	-
Kuvaus	Tulostaa listauksen, jossa on lueteltu kaikkien ladattujen portinlukijoiden tiedot. Listauksesta käy ilmi kunkin USB- tai sarjaportin osoite, niiden tiedonsiirtonopeudet, kuinka kauan ne ovat olleet päällä, kuinka kauan niitä aiotaan käyttää, minkä nimiseen lokitiedostoon niiden data tallennetaan ja mikä on kunkin lokitiedoston koko.

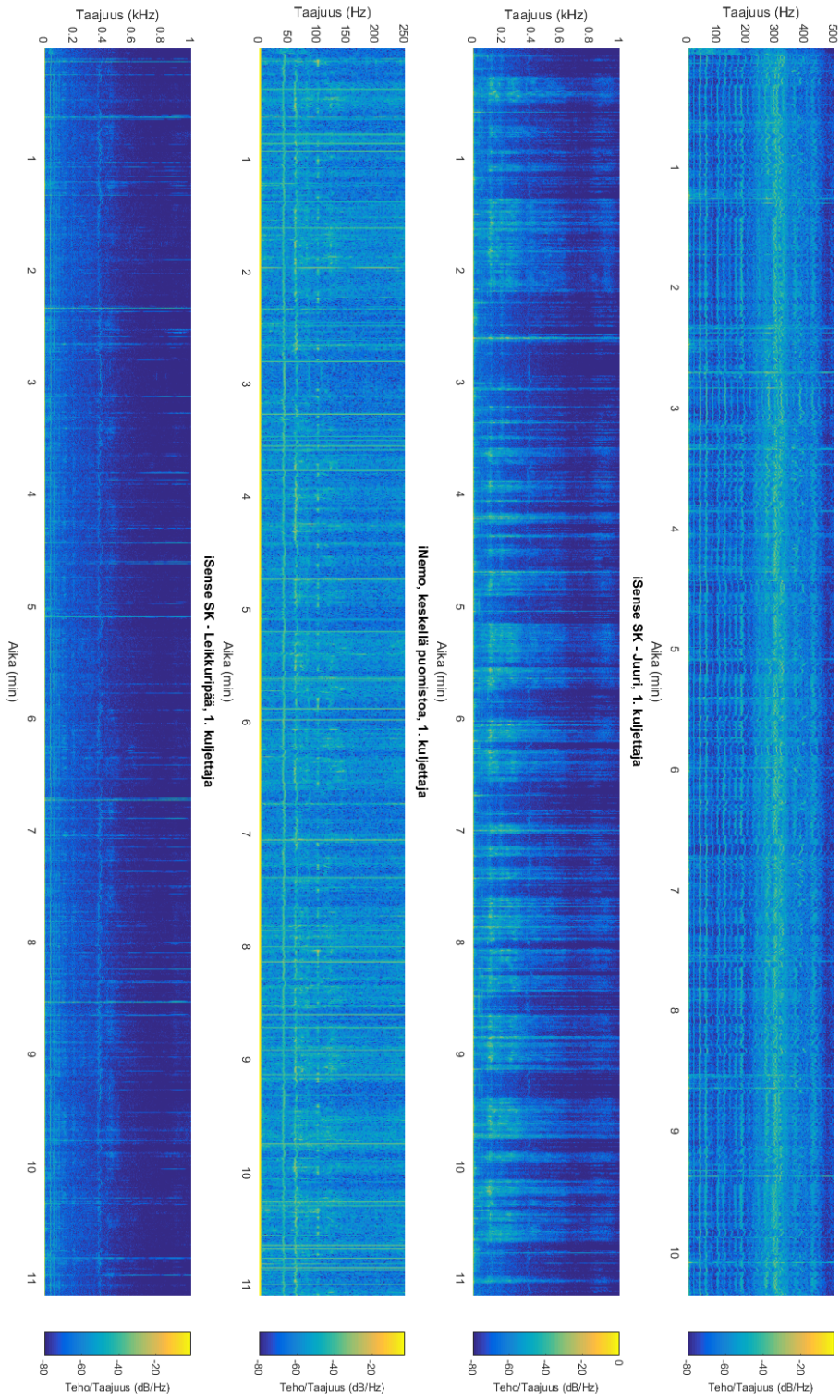
Ohjeet	
Komento	`help`
Parametrit	-
Kuvaus	Tulostaa rajapinnan ohjeet.

Yhteyden sulkeminen	
Komento	`quit`
Parametrit	-
Kuvaus	Eksplisiittinen komento-Bluetooth yhteyden sulkemiseksi. Yhteyden sulkeminen ei lopeta käynnissä olevaa mittausta. Yhteys voidaan katkaista myös ilman tätä komentoa.

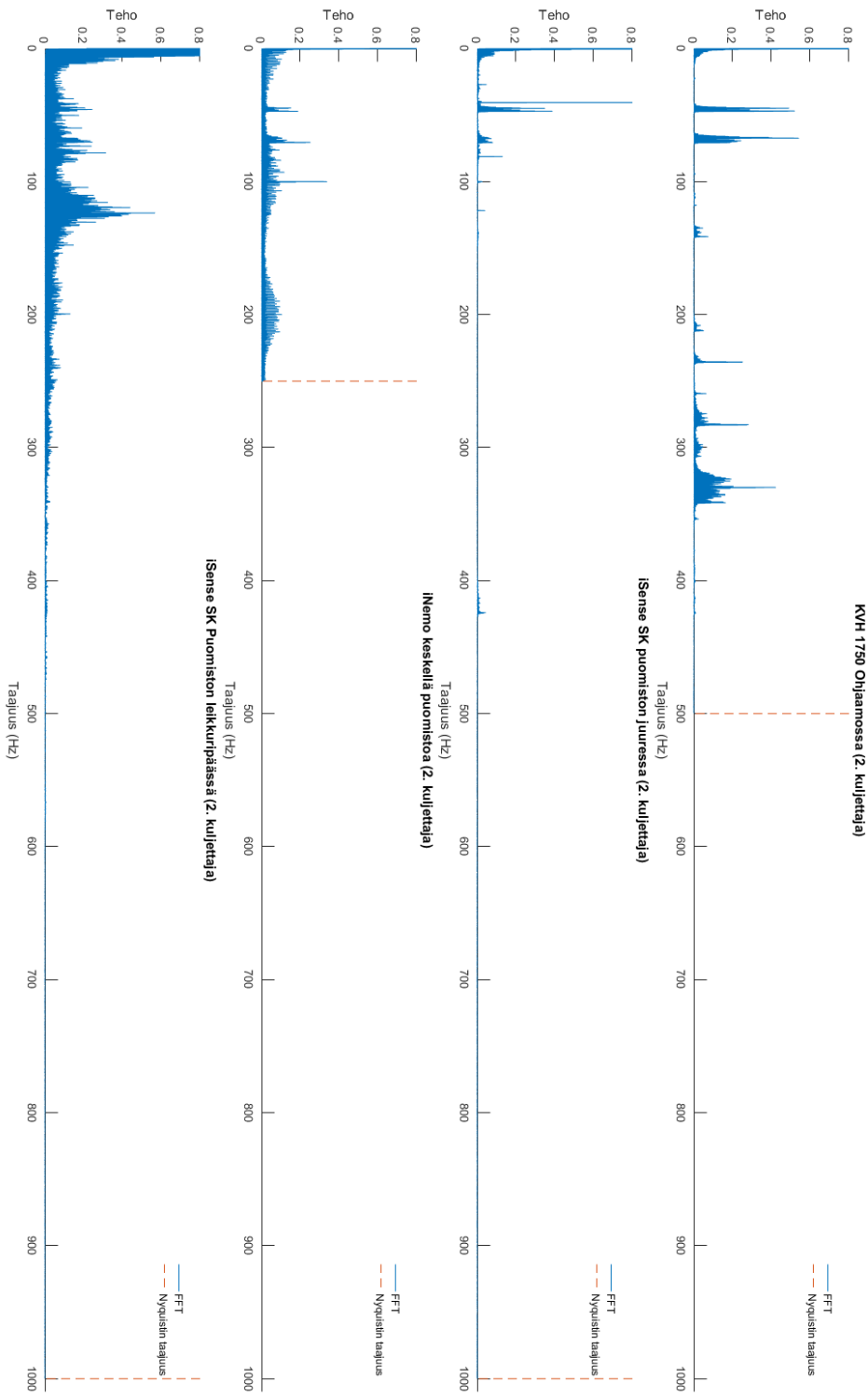
LIITE C. PIENMETSÄKONEMITTAUKSET



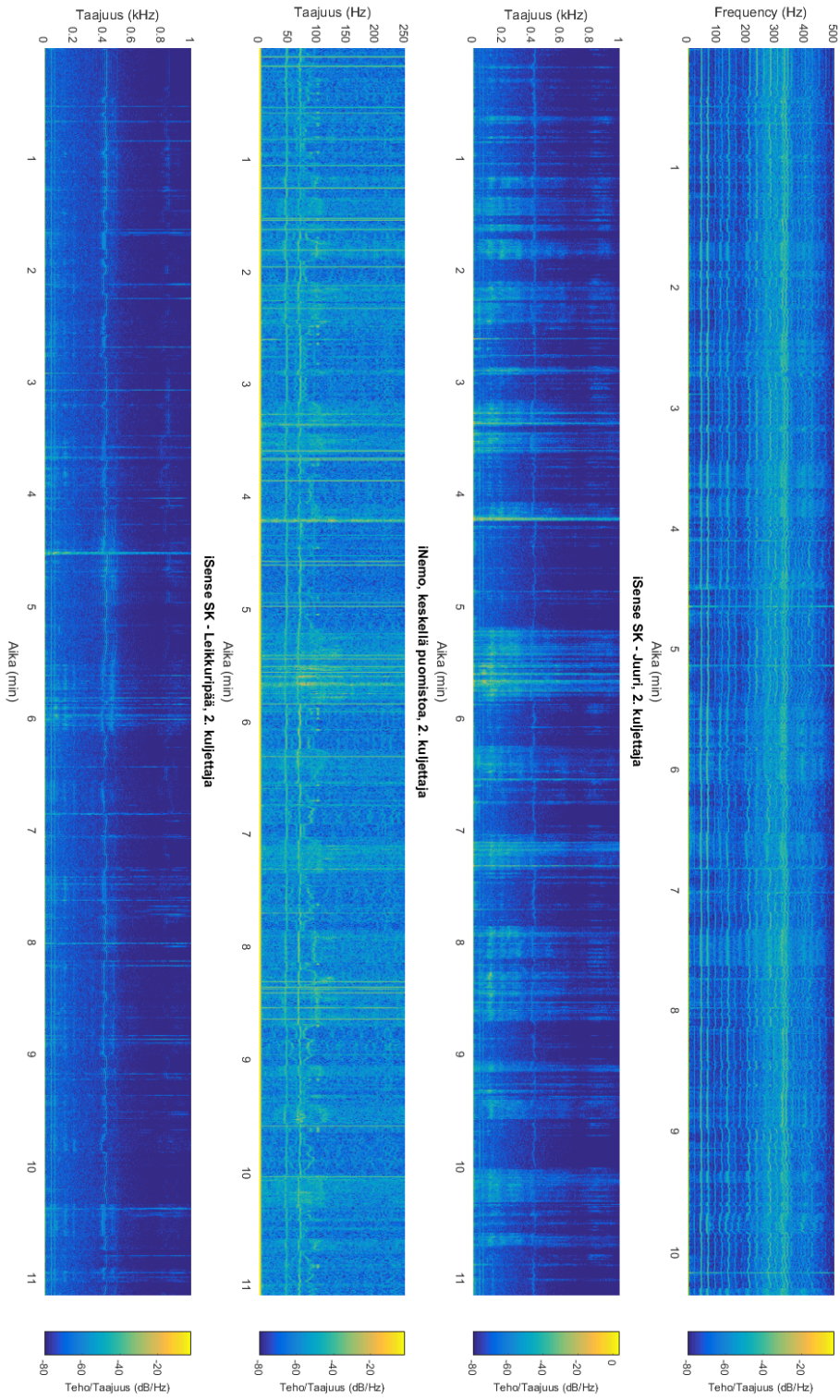
Kuva 1 *Ensimmäisen kuljettajan ajosta kerätyn datan fourier-muunnokset*



Kuva 2 *Ensimmäisen kuljettajan ajosta kerätyn datan spektrogrammit*



Kuva 3 *Toisen kuljettajan ajosta kerätyn datan fourier-muunnokset*



Kuva 4 *Toisen kuljettajan ajosta kerätyn datan spektrogrammit*